Universitat de Lleida

TREBALL FINAL DE GRAU

Estudiant: Albert Pérez Datsira

Titulació: Grau en Enginyeria Informàtica

Títol de Treball Final de Grau: **HealthSites, una app per a Android**

Director/a: **Montserrat Sendín Veloso**

Presentació

Mes: Setembre

Any: 2020

Sumari

Llista de Figures

Llista de Taules

1 Pròleg

Document on es presenta el projecte de l'aplicació HealthSites, una aplicació, redissenyada i renovada, desenvolupada amb *Kotlin* i *Firebase*, en la qual podem utilitzar el seu mapa integrat per veure en temps real els locals, tant restaurants com botigues, que ofereixen serveis vegans, vegetarians, ecològics, sense gluten i sense lactosa, juntament amb altres funcionalitats addicionals com el filtratge del mateix mapa.

A més a més, es pot deixar comentaris a cadascun dels perfils de les anomenades HealthSites i donar-li suport als més interessant deixant-los-hi un *like*, amb la finalitat de crear un registre de l'experiència dels usuaris. Així, qualsevol nouvingut podrà obtenir un *feedback* segons les opinions dels altres.

D'aquesta manera, els usuaris que desitgin menjar saludable o comprar productes específics, podran localitzar les HealthSites més properes a través de l'aplicació i decidir entre elles quina és l'adequada.

També, es pot seguir a les HealthSites per tenir les preferides llistades en la secció "Les meves HealthSites", la qual serveix de drecera per veure certa informació, deixar de seguir, realitzar comentaris i accedir fàcilment al perfil de cadascuna.

Finalment, hi ha un apartat central destinat únicament al registre de noves "HealthSites" perquè de fet, l'objectiu inicial serà aconseguir construir una base de dades (BBDD) per poder oferir les millors alternatives als usuaris.

Per concloure, m'agradaria agrair a la meva tutora del treball i professora de la Universitat de Lleida, Montserrat Sendín Veloso, que m'hagi introduït en el món del desenvolupament d'aplicacions per a dispositius mòbils, concretament en el món Android; i per haver-me fet més fàcil la realització del treball proporcionant-me comentaris sobre què i com millorar.

2 Introducció

En aquesta secció s'introdueix als lectors en el context del projecte amb la finalitat de facilitar-ne la comprensió.

Seguidament, s'explicaran els motius pels quals s'ha decidit realitzar aquest projecte, els objectius marcats, un petit resum del treball i l'estructura del document, en la qual s'introduirà breument cadascuna de les seccions.

2.1 Motivació

La idea de realitzar una aplicació centrada en aquest àmbit va sorgir conjuntament amb en Franco Friz durant l'assignatura Plataformes en Xarxa del 4t curs del Grau d'Enginyeria Informàtica de la Universitat de Lleida.

La nostra visió com a possible negoci futur, era obtenir una idea base la qual no intentés abraçar moltes funcionalitats ni voler fer moltes tasques ja que actualment en el mercat d'aplicacions per a dispositius mòbils hi ha molta varietat i un gran conjunt d'aplicacions multifuncionals contra les quals difícilment podríem competir.

Aleshores, volíem enfocar-nos en una tasca en concret què solucionés una necessitat real. Principalment, ho vam fer perquè la competència és molt forta i ja estaria assentada en el mercat, mentre que nosaltres estaríem començant. D'aquesta manera augmentàvem les possibilitats d'èxit.

Una vegada l'estratègia inicial a seguir estava decidida, vam realitzar un reconeixement del mercat per donar amb aquella tasca/funcionalitat que tingués un públic potencial. No buscàvem arribar a molta gent, perquè en el cas que solament s'arribés a l'1% de la gent que utilitza aplicacions mòbils actualment ja hauríem guanyat.

Aleshores va néixer la idea de fer una aplicació per tenir un registre de llocs saludables: veganes, vegetarians, sense lactosa, etc. És a dir per poder saber quins locals oferixen aquesta classe de serveis.

Aspectes positius de la idea? Doncs què no hi havia cap aplicació centrada únicament i exclusivament en aquesta funcionalitat, com també el constant creixement de la comunitat vegana, vegetariana, ecològica, etc, fet que fa augmentar els usuaris potencials.

2.2 Objectius

A continuació s'esmenten els objectius que es van plantejar a l'inici del projecte. Són les metes que es van marcar i sobre les quals s'ha anat treballant.

- Dissenyar una aplicació nova basada en la idea presentada prèviament
- Realitzar l'aplicació *Android* usant el llenguatge *Kotlin*
- Utilitzar *Firebase* per a la gestió d'usuaris
- Utilitzar *Firebase* per allotjar la BBDD
- Presentar un pla de negoci per a l'aplicació

2.3 Resum del treball

El document està destinat a explicar tots els àmbits del procés sobre la planificació, anàlisi, disseny i desenvolupament del projecte de software, concretament una aplicació *Android*.

El procés comença des de la planificació de tot plegat, continua amb l'anàlisi de viabilitat, passant pel disseny i la implementació i acaba amb les conclusions extretes i el teòric treball futur.

A més a més, s'ha realitzat un pla de negoci específic per a aquest projecte, i conjuntament amb el treball futur estableixen una camí a seguir.

Per acabar, en la següent secció es resumeix què conté cadascuna de les seccions més detalladament.

2.4 Estructura del document

Aquest aparta està destinat a narrar que hi ha en cada secció del document per ajudar a entendre i facilitar la cerca de continguts.

- **Pròleg**: resum inicial i presentació del document, a més d'incloure els agraïments al paràgraf final.
- **Introducció**: destinat a descriure el context i la motivació per realitzar el projecte, conjuntament amb els objectius marcats des d'un principi i amb un resum del treball.
- **Aspectes de planificació**: part es fan els primers estudis i planificacions, com ara els de la competència, sobre la viabilitat del projecte i introduint la metodologia utilitzada.
- **Anàlisi**: s'aprofundeix més en la definició dels requeriments i el seu estudi, mostrant i detallant els casos d'ús.
- **Descripció de la tecnologia utilitzada**: es fa menció a quines tecnologies s'han utilitzat, introduint-les i explicant-les detalladament.
- **Disseny**: secció dedicada a explicar el disseny i mostrar l'acabat final de la interfície.
- **Implementació**: secció on s'explica quines decisions importants s'han pres durant la implementació i es detalla certes parts específiques de la codificació.
- **Pla de negoci**: possibles accions per establir una estratègia de negoci.
- **Treball futur i Conclusions**: part final del document que es centra en com es podria millorar i què fe a partir d'ara, acabant amb les conclusions extretes.

3 Aspectes de planificació

En aquesta secció es troba la planificació del projecte. Una vegada es tenia clara la idea s'han de realitzar certs estudis per recaptar informació del context actual.

És molt important assentar les bases i establir el punt de partida. Entendre si és viable o no, si hi ha mercat o usuaris potencials, entre altres coses que són necessàries plantejar-se abans d'endinsar-se, i sobretot com es portarà a terme i quina metodologia de desenvolupament és l'adequada segons les circumstàncies.

Amb l'objectiu, clar, d'establir una ruta marcada amb garanties per dur a terme el projecte de software.

3.1 DAFO: Debilitats, Amenaces, Fortaleses i Oportunitats

L'anàlisi DAFO és una eina per realitzar un diagnòstic fiable i real sobre un projecte/empresa per prendre una decisió estratègica. Facilita la recollida d'informació valuosa en un context o mercat particular. En definitiva, serveix per veure les possibilitats d'èxit en el mercat en el qual es vol incidir.

Aleshores en funció dels resultats, es pot traçar un pla d'acció o pla de màrqueting si es decideix tirar endavant; o pel contrari adonar-se de la no viabilitat en el context actual.

Figure 1: Estructura DAFO

En l'anterior figura s'observa l'estructura del diagrama abraçant dues perspectives diferents. La primera es centra en un enfocament a nivell intern, en el qual s'ha d'enumerar i entendre quines són les debilitats i fortaleses pròpies. I la segona correspon a un enfocament extern, on s'estudia els elements relacionats amb el possible mercat i sector, remarcant les oportunitats i les amenaces.

Seguidament es mostra l'anàlisi sobre el projecte de l'aplicació a desenvolupar. A més a més, en la següent subsecció, s'incideix en l'anàlisi de la competència.

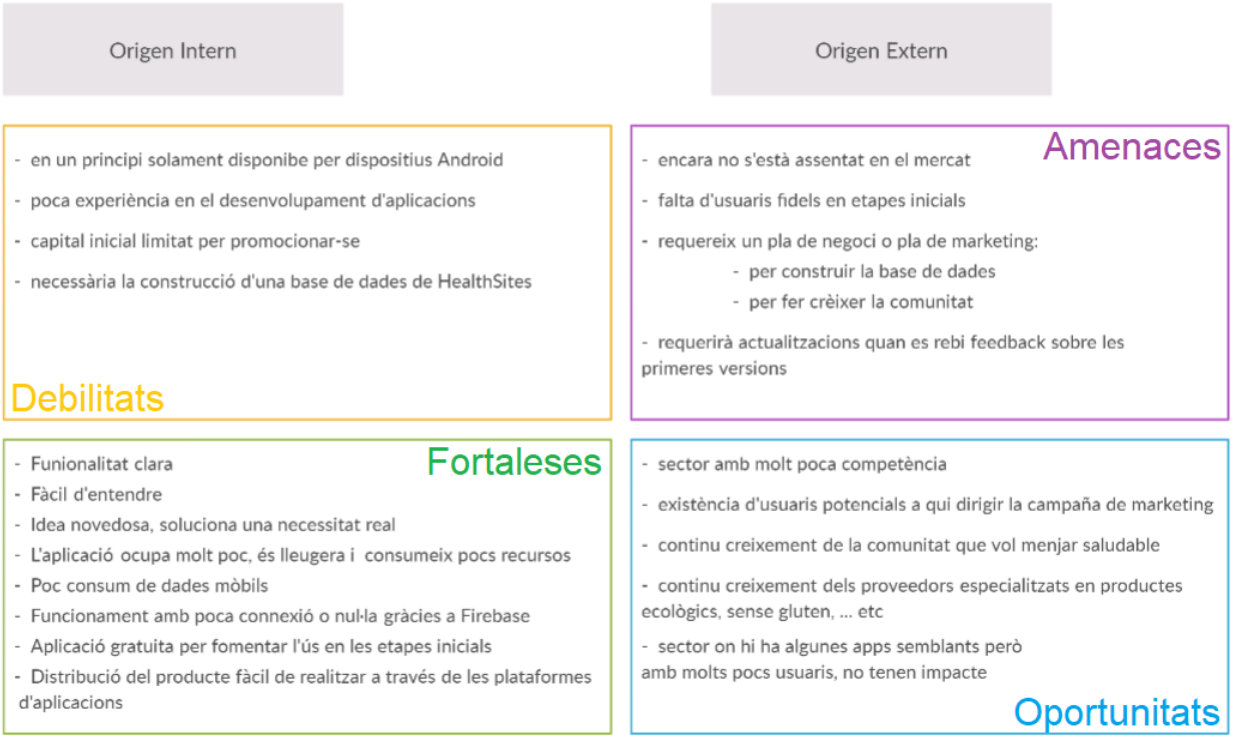


Figure 2: Anàlisi DAFO

3.2 Anàlisi de la Competència

Consisteix en una sèrie de mètodes investigatius que busquen prendre decisions estratègiques determinants en base als seus competidors comercials per poder superar-los i guanyar millors quotes de mercat. Clarament amb l'objectiu principal de posicionar-se líder en el respectiu sector.

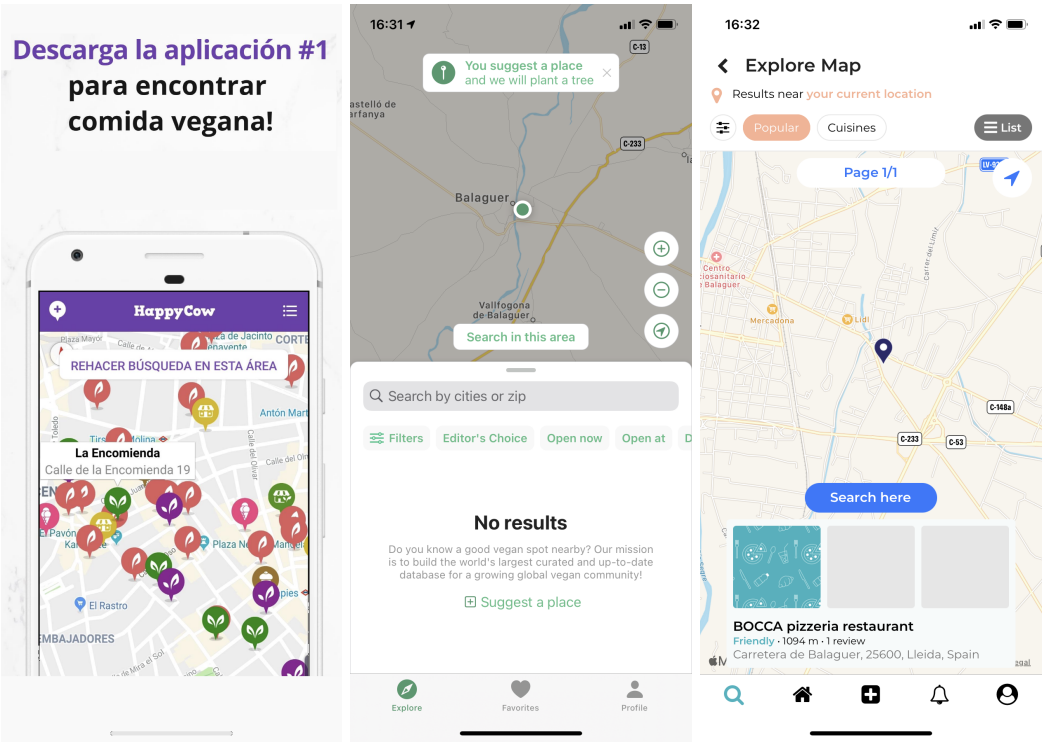
Per què és important? S'hi poden extreure moltes dades i avantatges competitives com les següents:

- Conèixer les estratègies comercials d'altres empreses
- Trobar necessitats en el mercat que no han estat ateses
- Entendre millor als clients potencials, tant pròpis com els de la competència
- Descobrir nous mètodes i maneres d'optimitzar processos
- Dissenyar tàctiques més creatives i eficaces
- Oferir experiències diferents als clients de les establertes fins ara

3.2.1 Identificació de competidors

Els principals competidors es troben als dos predominants mercats d'aplicacions: *Google Play Store* i *App Store*.

S'ha fet una cerca exhaustiva sobre les principals aplicacions que tracten una idea igual o semblant a la proposada, i se n'ha trobat tres, les quals tenen tant versió *Android* com *IOS*:



(a) HappyCow (b) VeganMaps (c) abillionveg

Figure 3: Pantalles dels Competidors

També s'ha trobat aplicacions sobre receptes veganes, vegetarianes, etc. Però no es tindran en compte a causa de la seva temàtica.

3.2.2 Anàlisi dels competidors

3.2.2.1 HappyCow

És una aplicació de pagament. A *Google Play Store* i a l'*App Store* hi trobem la versió de pagament, però en el primer cas també s'hi troba una versió gratuïta tot i que inclou pagaments interns. Realment no és el que sembla.

Té pocs usuaris, a pesar de tenir una antiguitat de +12 anys. És a dir, que no ha aconseguit en tot aquest temps de vida crear un impacte important. A més a més, disposa de pocs *ratings* però suficients comentaris negatius a tenir en compte.

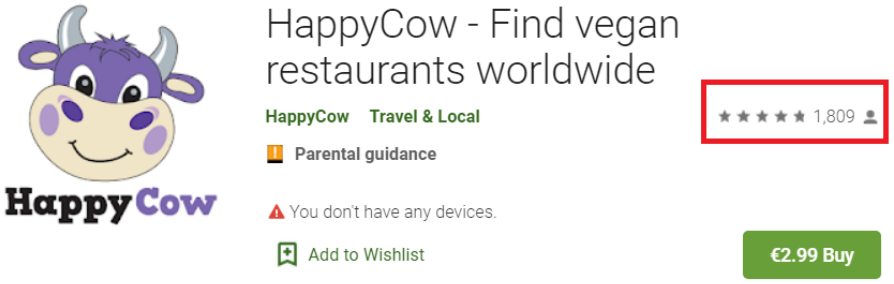


Figure 4: Rating de HappyCow

Té molts comentaris sobre problemes de funcionament i falles. Els usuaris també comenten que no és útil per a zones apartades de les grans ciutats, però fins i tot a les grans ciutats hi ha pocs llocs on anar.

Gairebé tots els comentaris tenen resposta per part dels desenvolupadors, cosa que significa que els importa el que pensi la gent. Esmenten que treballen per esmenar les falles i que la gent actualitzi l'aplicació si no ho ha fet.

També centren part de la funcionalitat de l'aplicació en crear una xarxa social per interactuar entre usuaris. Podria no ser la millor idea ja que s'ha de fomentar la interacció no entre usuaris, sinó cap a les *sites*.

Alguns comentaris:

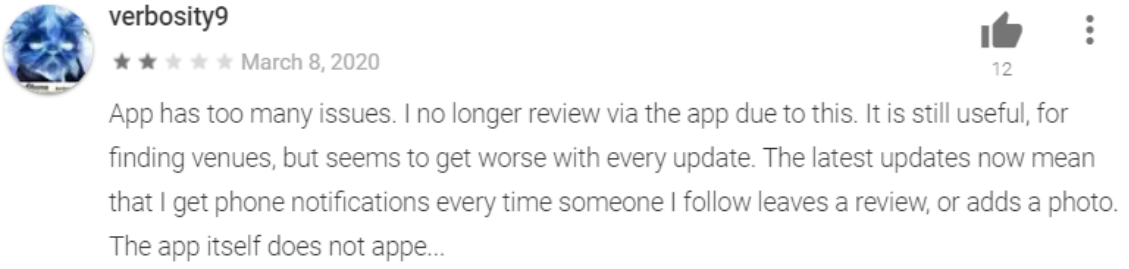


Figure 5: Comentari 1 de HappyCow

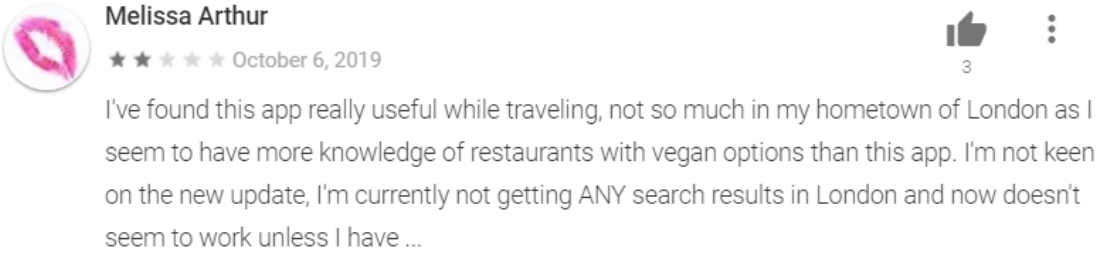


Figure 6: Comentari 2 de HappyCow

El primer usuari es queixa sobretot de la falta de llocs, que l'aplicació falli i que rebi moltes notificacions push, fet que sembla massa intrusiu. Cal destacar que és molt recent aquest comentari.

El segon, com he comentat anteriorment, troba que pot ser útil en els viatges però en la seva zona local no hi ha opcions. Però també comenta que amb la nova actualització no troba cap resultat a Londres.

Per acabar, l'aplicació disposa d'una pàgina web https://www.happycow.net/ però no està centrada en la promoció de l'aplicació. Es més a aviat una extensió o servei web, i el fet de no tenir cap mecanisme per aproximar-se a la gent els hi pot jugar en contra.

3.2.2.2 VeganMaps

Aquesta seria l'aplicació més semblant a la nostra idea. Però té encara menys usuaris que l'anterior; i sorprenentment no té cap tipus de comentari.

S'ha provat l'aplicació i no té llocs excepte en algunes grans ciutats. Tot i que no té cap punt de comparació en relació a la quantitat de llocs respecte el google maps.

És simple i fàcil d'utilitzar. El mapa està ben integrat. Per altra banda, es poden deixar *reviews* als llocs però no només carrega les pròpies de l'aplicació sinó que integra les de *Google Maps*. Cosa que pot produir que la gent es trobi comentaris totalment aliens a la temàtica.

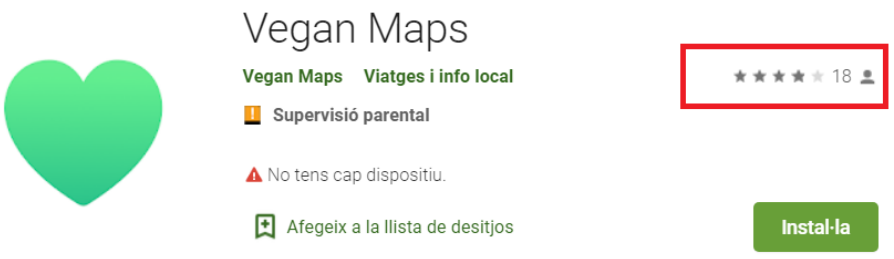


Figure 7: Rating de Vegan Maps

3.2.2.3 abillionveg

El mapa és secundari. És més aviat una xarxa social entre usuaris i reviews. És a dir, pots seguir usuaris per veure les seves *reviews*. Es centra bàsicament no en trobar llocs sinó en fer les citades *reviews* d'aquests. A part, tu com a usuari pots fer donacions per contribuir amb el desenvolupament de l'aplicació i els associats d'aquesta.

Té pocs usuaris, *ratings* i comentaris. Finalment pel que fa a la seva pàgina web https://www.abillionveg.com/ està molt ben feta i dona una visió positiva de l'aplicació introduint-te la seva funcionalitat.

Figure 8: Rating de abillionveg

No només és per a gent vegana, vegetariana, etc., com la nostra idea, sinó que tenen categories per a gent omnívora, entre altres.

Al mapa no hi ha ubicacions. Pots utilitzar un buscador però la llista de resultats no està al mapa. Deuen ser resultats trets directament de *Google Places* per exemple, i que no tenen relació amb la temàtica.

Per acabar, tot i els pocs comentaris aquests són força bons. Alguns exemples:

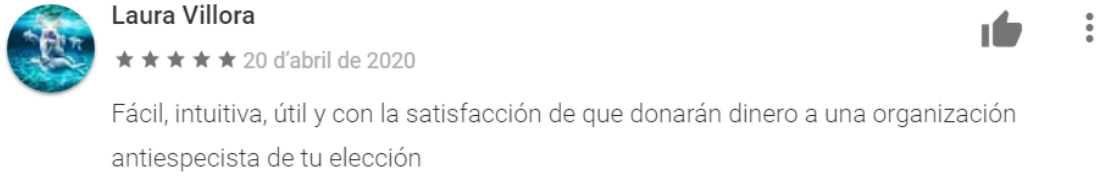


Figure 9: Comentari 1 abillionveg

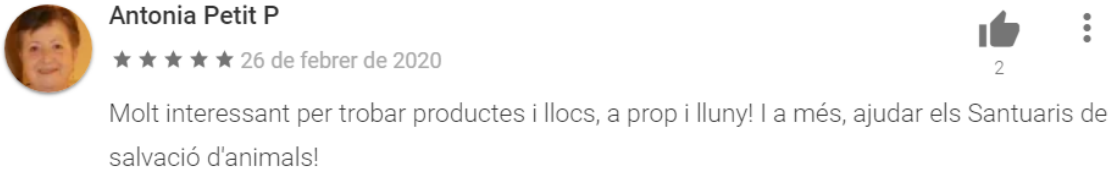


Figure 10: Comentari 2 abillionveg

3.2.3 Conclusions

No hi ha una clara competència, perquè aquestes aplicacions encara no tenen una gran quantitat d'usuaris fidels. Això juga a favor del nostre projecte.

La que més s'assembla és *Vegan Maps*. Les altres tenen altres funcionalitats i no es centren tant a trobar llocs específics, són més disperses. A *HappyCow* no pots deixar comentaris i, en canvi, *abillionveg* es centra en les *reviews* i no en les ubicacions, on pretén exercir més la funció de xarxa social entre usuaris.

Les campanyes de màrqueting, anuncis, etc. són inexistents. No s'ha trobat cap tipus de promoció, a part de les corresponents pàgines web. Tot i que l'única d'aquestes on es feia especial intenció de promocionar i explicar el funcionament de l'aplicació era la d'abillionveg.

En resum, és un molt bon sector en el qual establir un pla de negoci a través d'una nova aplicació. Aquesta ha de tenir la idea clara i no confusa de quina és la seva funcionalitat. *Vegan Maps* expressa molt bé aquests punts.

El pla de negoci hauria d'incloure accions per promocionar-se en el llançament de l'aplicació. A més a més, seria interessant la realització d'una pàgina web de suport i de promoció de l'aplicació. Com també realitzar la respectiva versió en *IOS*, igualment com els competidors, on inicialment es pot utilitzar conversors de ".apk" (extensió app per *android*) a ".ipa" (extensió app per *IOS*).

Si es realitza una pàgina web, a falta de competència clara seria relativament fàcil aconseguir bon posicionament en el buscador de *Google* amb keywords com: "vegan", "ecologic", "map", "app", etc.

Finalment, es creu que s'ha d'encarar l'aplicació a no utilitzar-la com xarxa social entre usuaris, sinó a aconseguir la interacció entre les ubicacions i els mateixos.

3.3 Metodologia

Es tracta d'un marc de treball utilitzat per estructurar, planificar i controlar el procés de desenvolupament del software.

Hi ha una gran quantitat de mètodes diferenciats pel seu enfocament i les seves característiques, per tant esdevindran més adients uns que uns altres segons les circumstàncies.

Per a aquest projecte utilitzar un **Model Evolutiu Incremental** sembla el mètode més adequat. El procés es divideix en 4 activitats o tasques principals: Anàlisi, Disseny, Codificació i Validació, i que per la producció de software s'inclouen en un treball en cadena o "*Pipeline*" formant un increment. Però aquestes no han estat aplicades de cop a tot el sistema sinó que han estat aplicades per separat sobre els diferents subsistemes o funcionalitats de què consta l'aplicació. És a dir, desenvolupant en paral·lel les diferents funcionalitats o subsistemes que s'aborden per separat els quals van sent assemblats entre sí. D'aquesta manera es va iterant, i al final de cada increment es rebrà *feedback* del client/usuari, considerant que al no haver un client definit (perquè és un TFG que sorgeix d'una iniciativa pròpia) es pot equiparar el client amb els usuaris finals, per aconseguir adaptar el software a les necessitats reals.

És un model que combina els avantatges dels corresponents seqüencial i iteratiu. Per una banda aplica repetidament el model seqüencial, i produeix a cada iteració un subconjunt de la funcionalitat del sistema (subsistema), anomenada increment.

"El sistema va creixent en cada iteració esdevenint en un increment fins arribar al producte final".

Es redueix considerablement el temps de desenvolupament perquè el model està en constant interacció amb el client sobre què s'ha pogut fer, què es podrà fer o allò que no s'ha pogut implementar. Per aquesta raó, gràcies a les entregues periòdiques de cadascun dels subsistemes no s'ha d'esperar a tenir una versió final del sistema i en conseqüència es pot anar refinant els requeriments com a resultat d'anar incrementat el mateix.

A més a més, aquest model és vàlid per a equips de desenvolupament reduïts, com és el cas. Ja que es desenvolupa el software pas a pas i testejant cadascuna de les noves funcionalitats, subsistema, o branca funcional, evolucionant i retroalimentant-se al llarg del procés.

La idea és de manera disciplinada anar modificant els requeriments i el disseny, incorporant noves branques funcionals a partir de noves idees o canvis a causa de limitacions del codi o serveis utilitzats que puguin sorgir. De fet, un dels grans avantatges és la capacitat de rebre *feedback* des del moment zero gràcies a involucrar directament el client, aconseguint un model eficient i adaptable als canvis.

Per acabar, en relació amb l'ús d'un model incremental les fases corresponents als diferents increments s'entrellacen durant el transcurs del desenvolupament, fet que facilita la interacció entre aquestes. És cert que els increments inicials es defineixen de bon principi i es prioritzen, però el que té de bo el model és que al compondre's per una cadena se'n poden anar incorporant de nous de manera creixent a partir de les conclusions extretes al final de cada fase/iteració del procés.

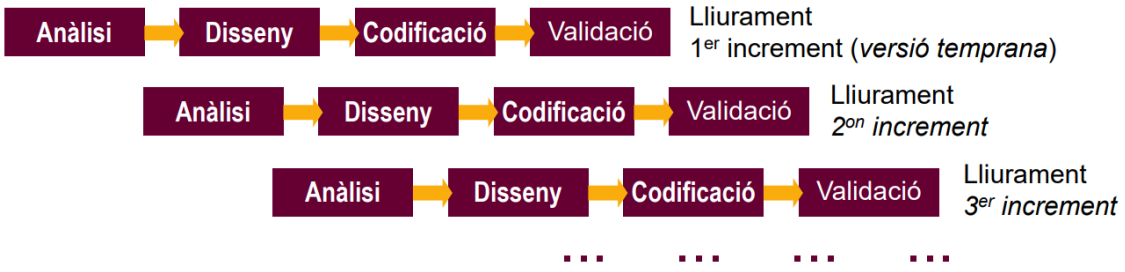


Figure 11: Diagrama del model evolutiu incremental

3.4 Requeriments per al desenvolupament del projecte

A continuació s'enumeren els requeriments tant de software com de hardware què es necessiten per portar a cap el projecte.

3.4.1 Requeriments de software

3.4.1.1 RSW1

Android Studio és l'entorn requerit per al desenvolupament del projecte, ja que l'entorn es centra únicament a facilitar precisament el desenvolupament d'aplicacions *Android*, per aquesta raó se'l considera el més adequat. Altrament, es pot optar per diverses alternatives com ara *Xamarin Studio*.

3.4.1.2 RSW2

Es necessita l'*Android SDK* (*Software Development Kit*), el kit de desenvolupament de software, el qual ens proporciona les eines per realitzar els nostres projectes *Android*.

3.4.2 Requeriments de hardware

3.4.2.1 RHW1

Per al desenvolupament de l'aplicació mòbil es requereix un ordinador/portàtil. S'aconsella utilitzar *Windows 10*, però també es podria utilitzar darreres versions, com la 8.1 o 8, tot i que ja estan desactualitzades. Versions del SO (Sistema Operatiu) més antigues ja no es recomanen perquè a partir de Windows 7, Microsoft ha deixat de donar suport. Altrament, es pot utilitzar *MacOS* també. En aquest cas s'ha usat un portàtil *MacBook Air* amb el SO *MacOS Mojave*.

3.4.2.2 RHW2

La computadora elegida ha de ser prou potent per poder emular els dispositius *Android* necessaris per realitzar les proves d'execució de l'aplicació. S'aconsella tenir un mínim de 8 gigabytes de RAM i un processador i5 de 4 cores. Justament, s'ha usat un portàtil amb 8 *gigabytes* (unitat de mesura) de RAM (Memòria d'Accès Aleatori o Memòria Principal) i un processador "i5" de vuitena generació amb 4 nuclis i 12 fils d'execució.

4 Anàlisi

En aquesta secció es presenta la fase d'anàlisi, una de les parts inicials del projecte, on es defineixen els requeriments del mateix establerts per al disseny i desenvolupament de l'aplicació amb una explicació el més detallada possible. S'ha d'entendre el projecte i l'objectiu d'aquest, les seves característiques i funcionalitats.

En resum, es comencen a llençar idees per dividir el sistema en branques funcionals que puguin ser útils.

4.1 Requeriments funcionals

Els requeriments funcionals són les descripcions explícites del comportament que ha de tenir una solució de software i quina informació ha de gestionar. Han de proporcionar una descripció suficientment detallada per permetre el desenvolupament i implementació de la solució.

La seva importància radica en entendre què la gestió dels requeriments funcionals es citada com una de les causes més freqüents que ocasionen insatisfacció de les expectatives. Per tant, s'ha de tenir especial cura i no generar descripcions ambigües produint interpretacions errònies.

4.1.1 RF1

L'aplicació disposarà d'un mapa on els usuaris podran veure les HealhSites afegides.

4.1.2 RF2

Els usuaris han de poder deixar la seva opinió en forma de comentaris sobre una HealthSite en concret.

4.1.3 RF3

Cadascuna de les HealhtSites haurà de tenir un perfil, on s'hi mostri la seva informació de contacte, els comentaris dels usuaris, els seus oferiments/propietats, els seus seguidors i nombre de comentaris fins al moment, entre altres dades.

4.1.4 RF4

Els usuaris hauran d'autenticar-se per entrar dins de l'aplicació.

4.1.5 RF5

S'haurà d'afegir un formulari de registre per als nous usuaris que es vulguin unir a la comunitat.

4.1.6 RF6

S'haurà d'afegir un formulari per al registre de noves HealthSites.

4.1.7 RF7

L'usuari disposarà del seu perfil per veure les seves dades, i poder-les modificar si ho desitja. A més a més podrà veure el nombre de HealthSites que està seguint fins al moment.

4.1.8 RF8

L'usuari disposarà d'una llista de les HealthSites que segueix actualment, per poder accedir ràpidament al seu perfil o realitzar accions ràpides com ara deixar-la de seguir.

4.1.9 RF9

L'aplicació tindrà un apartat de filtres sobre el mapa, on es podrà marc quin tipus de HealthSites es desitgen veure en el mapa.

4.1.10 RF10

L'aplicació disposarà d'un apartat d'ajustos per realitzar diverses accions, com canviar el llenguatge, tancar sessió, etc.

4.1.11 RF11

L'aplicació disposarà d'una pantalla per accedir a les opcions de personalització i configuració del mapa.

4.1.12 RF12

L'aplicació disposarà d'una pantalla on es podrà compartir o reportar una HealthSite en concret si l'usuari ho desitja.

4.2 Requeriments no funcionals

Els requeriments no funcionals són qualitats o atributs del software que ha d'exhibir un sistema en realitzar la seva funció i quines restriccions ha de respectar.

No estan directament associats amb el que fa el software. És més, sense ells el sistema també funcionaria, però no n'hi ha prou. Seguidament s'enumeren classificant-los en les seves categories.

4.2.1 Producte

4.2.1.1 RNF1: Usabilitat

S'ha d'aplicar el DCU (Disseny Centrat en l'Usuari) per aconseguir una interfície senzilla, intuïtiva i usable.

4.2.1.2 RNF2: Hardware

L'aplicació necessita que es faci ús de la càmera si l'usuari vol actualitzar la seva imatge de perfil o si es vol afegir una imatge representativa a la nova HealthSites que s'està creant.

4.2.1.3 RNF3: Hardware

L'aplicació necessitarà accés a l'emmagatzematge intern del dispositiu mòbil si es vol recuperar alguna fotografia per utilitzar-la en l'aplicació.

4.2.1.4 RNF4: Hardware

L'aplicació farà ús de la memòria del dispositiu per guardar les preferències de l'usuari a l'hora d'usar i configurar els paràmetres de l'aplicació.

4.2.1.5 RNF5: Hardware

L'aplicació necessitarà accedir a la localització del dispositiu per mostrar-la en el mapa i poder establir un seguiment a mesura que el dispositiu es mou.

4.2.1.6 RNF6: Hardware

L'aplicació necessitarà accedir a Internet per comunicar-se amb la base de dades respectiva.

4.2.1.7 RNF7: Portabilitat

Al ser una versió nova es considera posar el nivell de l'API (*Application Programming Language*) mínim a la versió 29.

4.2.1.8 RNF12: Mantenibilitat

El sistema ha de ser fàcilment modificable en el menor temps possible per poder reparar errors o falles inesperades.

4.2.1.9 RNF13: Implementació

Les dades de l'aplicació hauran d'estar emmagatzemades en un sistema gestor de BBDD, sobre el qual es puguin realitzar futures consultes no previstes actualment.

4.2.2 Organització

4.2.2.1 RNF8: Llicenciament de software

La totalitat dels components i llibreries que s'utilitzin en el desenvolupament han de ser de programari lliure amb llicència BSD (*Berkeley Software Distribution*) o GLP (*GNU General Public License*).

4.2.3 Extern

4.2.3.1 RNF9: Privacitat

El sistema no podrà revelar cap informació personal dels clients a part del seu usuari que no hagi estat permesa per ells mateixos, complint el RGPD (Reglament General de Protecció de Dades).

4.2.3.2 RNF10: Seguretat

Les dades de l'aplicació només podran ser modificades per aquelles persones autoritzades per la mateixa. Solament podran llegir i escriure de la BBDD els usuaris autenticats com a tal.

4.2.3.3 RNF11: Seguretat

El sistema ha de mantenir la consistència de la BBDD i assegurar-se de no tenir certes dades duplicades com ara els correus electrònics per iniciar sessió.

4.3 Especificacions de casos d'ús

En aquest apartat es mostrarà el diagrama de casos d'ús de l'aplicació realitzat per representar gràficament com l'usuari interaccionarà amb el sistema.

4.3.1 Diagrama de casos d'ús

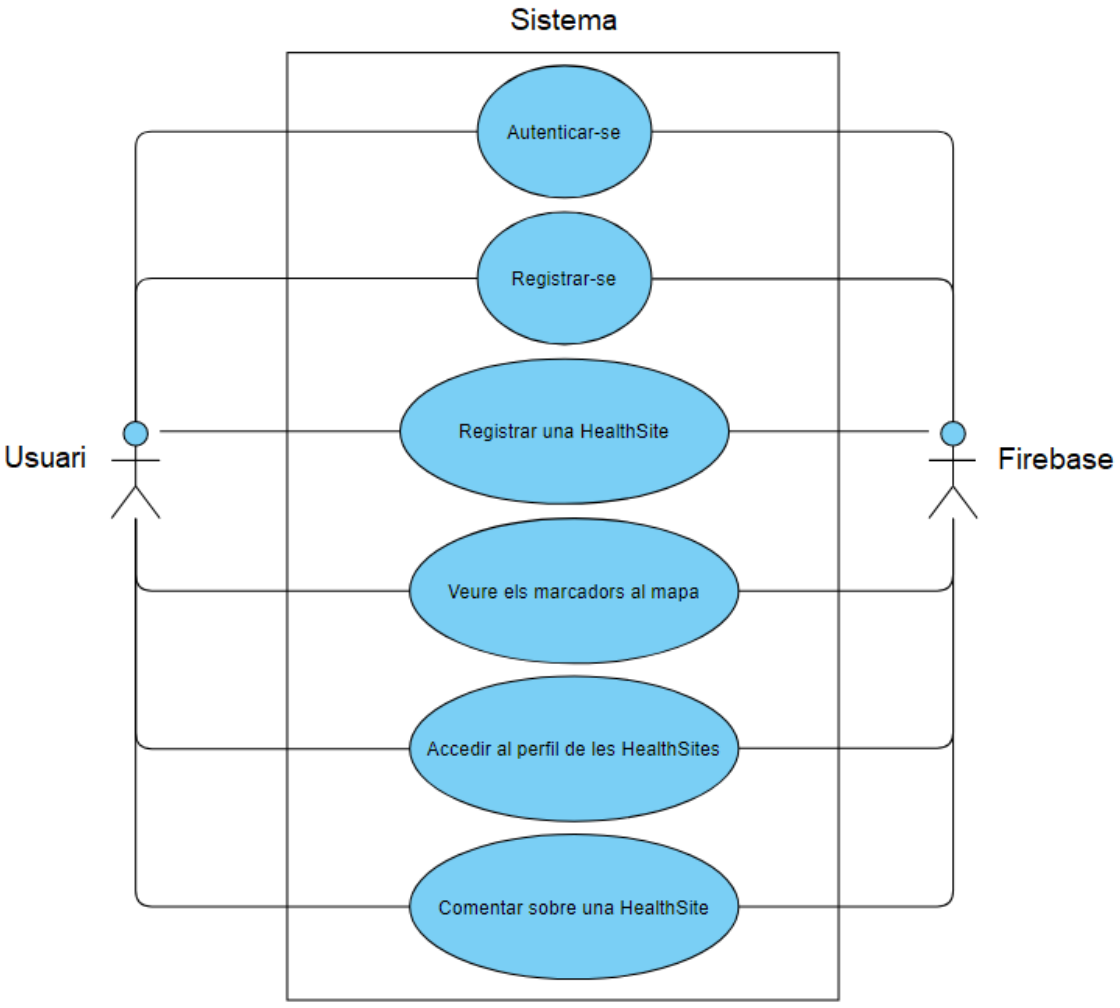


Figure 12: Diagrama de casos d'ús

El diagrama ens mostra els casos d'ús base de l'aplicació, en els quals recau la funcionalitat principal d'aquesta. A mesura que s'ha anat avançant en el desenvolupament i disseny posterior de l'aplicació, s'han afegit altres funcions com la de seguir, donar *like* als comentaris, entre altres.

En el següent punt es descriurà cadascun dels casos d'ús del diagrama per a una millor comprensió.

4.3.2 Casos d'ús

CU-1	
Cas d'ús	Autenticació d'usuari
Actors	Usuari de l'aplicació
Objectiu	Accedir a l'aplicació per poder-la utilitzar
Precondicions	Tenir un compte ja creat
Escenari	S'introdueix l'email i la seva contrasenya.
Postcondicions	Es procedeix a la verificació del compte i de les credencials. En cas afirmatiu, es permet l'entrada a l'interior de l'aplicació.
Excepcions	L'email ha de tenir el format correcte. La contrasenya ha de tenir almenys 6 dígits. En cas d'identificació errònia, es mostrarà un missatge a l'usuari.

Table 1: Primer cas d'ús

CU-2	
Cas d'ús	Registre d'usuari
Actors	Usuari de l'aplicació
Objectiu	Crear un nou compte
Precondicions	Tenir l'aplicació descarregada i instal·lada al dispositiu
Escenari	S'accedeix a l'aplicació i es prem el botó per començar el registre del nou compte. S'introdueixen les dades que es demanen.
Postcondicions	Es procedeix a la verificació de les dades i a la creació del compte.
Excepcions	En les respectives pantalles del formulari, hi haurà restriccions sobre el tipus de dades permeses en cadascun dels camps. Per exemple, en l'email només s'acceptaran determinats formats. Altrament, si no es pot crear el compte perquè ja està en ús s'informarà a l'usuari.

Table 2: Segon cas d'ús

CU-3	
Cas d'ús	Registre HealthSite
Actors	Usuari de l'aplicació
Objectiu	Crear una nova HealthSite
Precondicions	Tenir un compte i estar loguejat.
Escenari	S'accedeix a l'aplicació i es prem el botó central de l'anomenada *Bottom Navigation* (Barra de navegació inferior) situada en la pantalla principal. Finalment s'introdueixen les dades al formulari.
Postcondicions	Es procedeix a la verificació de les dades i a la creació de la nova HealthSite.
Excepcions	En les respectives pantalles del formulari, hi haurà restriccions sobre el tipus de dades permeses en cadascun dels camps. Altrament, si no es pot crear s'informarà a l'usuari.

Table 3: Tercer cas d'ús

CU-4	
Cas d'ús	Veure els marcadors de les HealthSites en el mapa
Actors	Usuari de l'aplicació
Objectiu	Recuperar la informació de la base de dades i mostrar els marcadors al mapa, juntament amb altra informació.
Precondicions	Tenir l'aplicació descarregada i instal·lada al dispositiu, i estar loguejat.
Escenari	S'accedeix a l'interior de l'aplicació a l'apartat del mapa.
Postcondicions	
Excepcions	En cas que falli la connexió es mostrarà un missatge a l'usuari sobre la pèrdua d'aquesta, cosa que implicaria no poder descarregar les dades de les HealthSites per mostrar els marcadors respectius.

Table 4: Quart cas d'ús

CU-5	
Cas d'ús	Veure informació de la HealthSite
Actors	Usuari de l'aplicació
Objectiu	Veure les característiques, contacte, *feedback* dels usuaris, etc; de les HealthSites.
Precondicions	Tenir l'aplicació descarregada i instal·lada al dispositiu, i estar loguejat.
Escenari	S'accedeix a l'aplicació i es prem el marcador del mapa de la HealthSite de la qual volem saber més informació. Se'ns mostrarà una *Info Window* del marcador amb certa informació reduïda. A més a més podem prémer la *Info Window* per accedir a tota la informació completa de la HealthSite obrint el seu perfil.
Postcondicions	
Excepcions	Pot donar excepcions qualsevol tipus de falta de connexió.

Table 5: Cinquè cas d'ús

CU-6	
Cas d'ús	Comentar sobre la nostra experiència en una HealthSite
Actors	Usuari de l'aplicació
Objectiu	Donar la nostra opinió en format de comentari.
Precondicions	Tenir l'aplicació descarregada i instal·lada al dispositiu, i estar loguejat.
Escenari	S'accedeix a l'aplicació, i concretament al perfil específic de la HealthSite. Aleshores es prem el botó de "Comentar", s'introdueix el missatge i es confirma.
Postcondicions	S'actualitza la base de dades tot afegint el nou comentari, i es sincronitza amb les aplicacions client.
Excepcions	Qualsevol tipus de falta de connexió pot esdevenir en una excepció. Tot i que si el comentari no es pot pujar en el moment, es guarda en la memòria cau i posteriorment ja es pujarà.

Table 6: Sisè cas d'ús

5 Descripció de la tecnologia utilitzada

5.1 Entorn de desenvolupament

L'aplicació s'ha realitzat mitjançant l'entorn de desenvolupament integrat (IDE) *Android Studio*, que es centra en el desenvolupament d'aplicacions. Vaig elegir usar aquest IDE per la seva capacitat d'oferir moltes funcions i integritat que ajuden al desenvolupador a ser més productiu i a centrar-se en l'aplicació i no tant en la configuració del projecte en si. A més a més, té disponible un emulador en el qual es poden testejar ràpidament i fàcilment les versions de la nostra aplicació sense haver d'usar un dispositiu físic o un emulador extern.

Altrament, utilitza un sistema de compilació basat en *Gradle*. És una eina de compilació integrada, la qual ens dóna rendiment i flexibilitat al despreocupar-nos de la gestió de les dependències i fins i tot ens permet usar altres tipus de llenguatges diferents de Java, com el cas de *Kotlin*. A més a més ens ofereix la possibilitat de comparació de *builds*, en la qual podem observar les diferències en el cas d'obtenir algun error. D'aquesta manera podem analitzar la causa i resoldre-la, a part que facilita la integració amb repositoris de projectes i amb *Firebase*.

5.2 Llenguatge utilitzat

Pel que fa al desenvolupament de l'aplicació, s'ha realitzat usant *Kotlin*, un llenguatge de programació orientat a objectes i estàticament tipat, desenvolupat per *JetBrains* a partir del 2010.

És cert que és un llenguatge oficial per desenvolupar aplicacions *Android*; però també es pot utilitzar en altres àmbits, com a llenguatge servidor, en llocs web o en el sistema operatiu d'Apple (IOS).

Una de les parts més atractives és la seva total compatibilitat amb Java, la qual esdevé gràcies al fet que s'executa en la Màquina Virtual de Java (JVM). Va ser dissenyat específicament per a aconseguir la interoperativitat amb Java, tot i que també ho és amb *Javascript*.

Aleshores, podem tenir mòduls de la nostra aplicació en Java que perfectament es comuniquin amb les altres parts del codi fetes amb *Kotlin*. Per això, qualsevol llibreria o dependència en Java es podria usar en un projecte amb *Kotlin*, fet que fa més usable i còmode aquest llenguatge.

Per altra banda, durant els darrers dos anys s'ha avançat molt en migrar les llibreries de Java a *Kotlin*, i s'ha donat molt suport en la realització de la documentació necessària per al seu ús sobretot des que *Google* en la *Google* I/O de 2017 informava que donaria suport a partir d'*Android* 3.0, al desenvolupament en *Kotlin*, cosa que va esdevindre en un continu creixement de la comunitat de desenvolupadors. No obstant això, si una llibreria/dependència Java encara no s'ha migrat no hi haurà cap problema en usar-la.

Al corre sota la JVM permet que aquesta de forma dinàmica estableixi el tipus de les variables, i aleshores no cal especificar-ho en la codificació.

En relació amb la seva codificació, aquesta és molt neta i reduïda. Usant *Kotlin*, s'arriba a reduir de forma significativa les línies de codi, amb la qual cosa no només els fitxers estaran més simplificats sinó que l'aplicació no pesarà tant.

5.3 Serveis utilitzats

S'ha usat *Firebase* com a sistema de base de dades. En termes generals, *Firebase* és una plataforma desenvolupada per a la creació d'aplicacions mòbils i web.

Ofereix molts serveis, els quals han anat augmentant durant els anys. Els principals són la *Realtime DataBase*, el *Firebase Authentication*, el *Firebase Firestore*, *Cloud Storage*, *Cloud Functions*, etc., entre molts altres.

En concret, en aquest projecte s'ha usat *Firebase Authentication*, *Firebase Firestore*, i *Cloud Storage*.

Firebase Authentication ens permet aconseguir una integració dinàmica dels usuaris. Ens facilita les tasques d'autentificació i verificació dels usuaris; en definitiva, podem accedir a totes aquestes funcionalitats a partir de la seva API.

Firebase Firestore o *Cloud Firestore* és una base de dades flexible i sobretot escalable, característica principal que la diferència de la seva germana *Realtime Database*. Està disponible per a la programació amb servidors, dispositius mòbils i web. A més a més també manté les dades sincronitzades antre aplicacions client a través de *listeners/agents* per escoltar els canvis a temps real.

A part d'això, també ofereix assistència sense connexió per als dispositius mòbils i web, i per aquest motiu no importa la latència de la xarxa ni la connectivitat ja que s'està preparat per gestionar aquests casos. *Cloud Firestore* manté en la memòria cau del dispositiu les dades que utilitza la nostra aplicació, i així és com es pot llegir, escoltar, escriure sense connexió, ja que quan es torna a estar connectat, *Cloud Firestore* sincronitza els canvis locals amb les dades del núvol.

Concretament és una base de dades *NoSQL* (no relacional) orientada a documents allotjada al núvol, que emmagatzema les dades en Col·leccions de documents en format JSON (*JavaScript Object Notation*). En definitiva, manté una jerarquia en la qual es van anidant documents. Cada document conté una llista de parells clau - valor, els quals poden contenir tant subcol·leccions com objectes compostos.

Les consultes per recuperar les dades són més àmplies que a la *RealTime Database*. Pots realitzar tant consultes simples com compostes, recuperar solament un document o diversos, utilitzar filtres específics per a camps en concret, i fins i tot aplicar una ordenació als resultats.

Per acabar, *Cloud Storage* és un altre servei de *Firebase* que es va crear per a aquelles aplicacions que necessitin emmagatzemar i recuperar contingut generat pels usuaris, com fotos o vídeos, l'emmagatzematge dels quals no és viable amb les bases de dades no relacionals usades fins ara.

És un servei simple i potent, el qual mitjançant el SDK de *Firebase* per *Cloud Storage* s'afegeix la seguretat de *Google* a les operacions de càrrega i descàrrega de fitxers, sense importar la qualitat de la xarxa.

Igualment com es va fer amb *Firebase Firestore*, *Cloud Storage* va ser dissenyada per aconseguir una gran escalabilitat ja que es pensava en els futurs requeriments de les aplicacions. Aleshores si en el cas que la nostra aplicació assoleixi l'èxit, el sistema ja estaria preparat per suportar aquest gran canvi.

Finalment, totes les dades guardades es poden protegir usant *Firebase Authentication* i les pròpies regles de seguretat tant de *Firebase Firestore* com de *Cloud Storage* respectivament.

6 Disseny

6.1 Arquitectura del sistema

6.1.1 Arquitectura Física

En aquest apartat s'explicarà l'arquitectura del sistema. Recalco el sistema ja que no només està format per la pròpia aplicació que esdevindria la part del client anomenada el *front-end*, sinó que tenim un altre actor que correspon al servidor de *Firebase*, formant la part del *back-end*.

El *front-end* i el *back-end* són dos termes que es refereixen a la capa de presentació i a la capa d'accés a les dades, respectivament. El *front-end* correspon a la interfície de l'usuari, la part tangible a partir de la qual l'usuari interactua. I el *back-end* és la part del servidor, encarregat d'enllaçar el client amb l'accés a dades i la resta de funcions.

El sistema en la seva globalitat segueix una arquitectura basada en client-servidor. L'aplicació i el servidor es comuniquen per autenticar usuaris, emmagatzemar dades, recuperar o sincronitzar els canvis, o guardar les dades generades pels clients.

Seguidament es presenta el diagrama de xarxa per representar l'arquitectura física de forma gràfica.

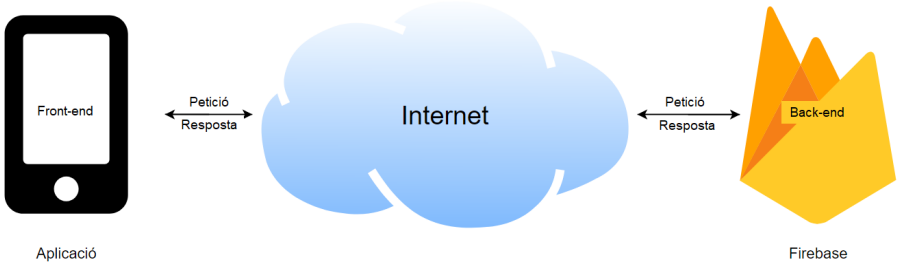


Figure 13: Arquitectura física del sistema

En la figura anterior, es pot observar l'arquitectura. L'aplicació i el servidor de *Firebase*, l'encarregat de la gestió de la base de dades i de l'autenticació dels usuaris, estan en comunicació constant a través d'Internet.

El client fa peticions al servidor per rebre una resposta, i també estableix *callbacks* per aconseguir una sincronització de dades en temps real.

6.1.2 Arquitectura del servei

Partim del punt on una aplicació *Android* es basa en renderitzar les pantalles a partir de l'estructura definida pels fitxers XML (*Xtensible Markup Language*) i inflar-los segons les dades oportunes.

Llavors, la pròpia manera de treballar del *framework* ja et manté una clara separació entre les vistes de la lògica de negoci, encarregada d'enllaçar amb les dades. Implícitament ja s'està aplicant el paradigma Model Vista Controlador (MVC) basat en el principi de separació d'interessos.

Per una banda, les vistes, com ja s'ha comentat, recauen sobre els fitxers XML que estableixen la seva estructura. Per altra, la lògica de negoci disposa el renderitzat de les vistes, com es comporten, etc. i també tot allò relatiu a què fer quan l'usuari interactua amb l'aplicació. Això comporta cridar al model per realitzar les accions requerides.

Es pot optar per utilitzar les Activitats o Fragments com a controladors si realment són petites les tasques sobre el model. Tot i que en casos on hi ha una gestió gran també es pot externalitzar el controlador a una classe específica, la qual seria instanciada per l'Activitat i així no fer-la tant dependent de la plataforma.

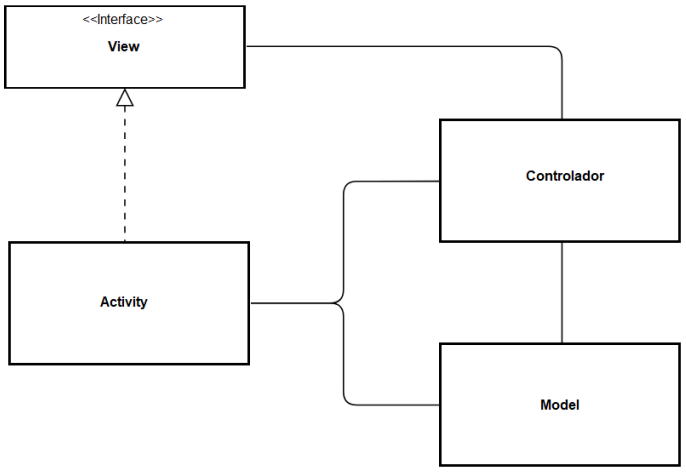


Figure 14: Diagrama de capes

6.1.3 Arquitectura Lògica

Pel que fa a l'aplicació *Android*, aquesta es basa en una separació entre la capa de presentació, els fitxers XML, la qual renderitzada per la capa de negoci, és a dir, el propi codi intern de l'aplicació; i la capa d'accés a les dades formada pel *back-end*.

- Capa de presentació: nivell superior de l'aplicació. Proporciona una interfície per a l'usuari. La formen tots els elements que capten la interacció/informació del usuari. Es comunica amb la capa de negoci.
- Capa de negoci: conté els processos a realitzar a partir de la informació rebuda des de la capa de presentació. És l'intermediari entre la capa de presentació i la capa d'accés a les dades. La lògica interna.
- Capa d'accés a les dades: on es gestionen i es guarden les dades. S'encarrega de l'accés a les dades per fer consultes, actualitzacions, emmagatzematge o eliminació d'aquestes. Es comunica amb la capa de negoci, facilitant les dades o rebent-les.

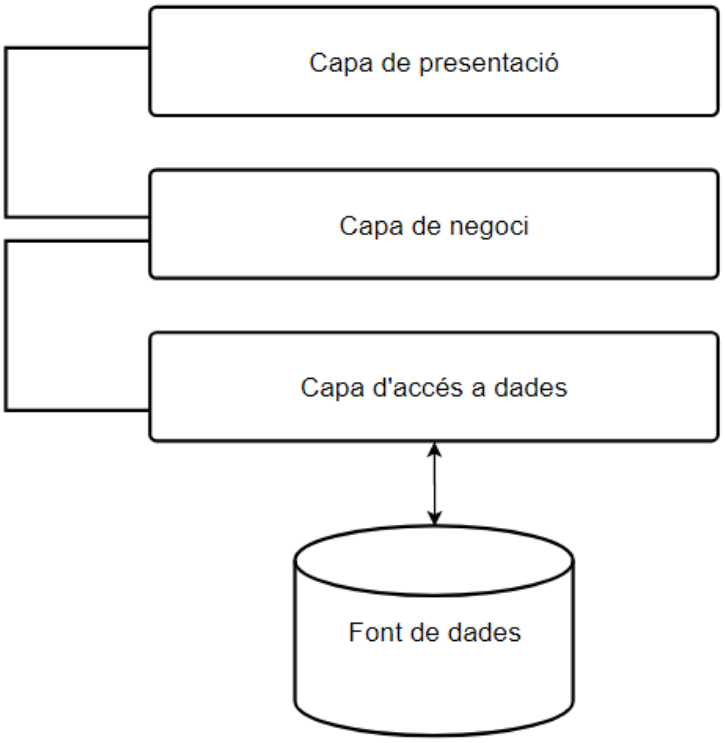


Figure 15: Diagrama de capes

Internament es treballa amb models de dades per representar les entitats "HealthSite" i "User", les quals formarien la capa de dades o del model de dades. Es va optar per no representar el "Post" o comentari sobre una "HealthSite" com una entitat en el codi, a causa que solament es mostren les seves dades sense tenir cap interacció/relació amb altres parts del codi; tot i que conceptualment s'ha contemplat.

6.2 Model de dades

6.2.1 Diagrama UML de les entitats base

En el títol de la subsecció, esmento que el següent diagrama UML (*Unified Modeling Language*)serà el format per les entitats base, és a dir, les que marquen l'estructura del conjunt de l'aplicació, ja que hi ha mòduls i llibreries les quals donen suport per a algunes accions en concret però no pertanyen directament a l'estructura base del disseny de l'aplicació.

Les entitats i com es relacionen aquestes es pot extrapolar per implementar-ho en altres projectes en plataformes i *frameworks* diferents però la part pertanyent a l'entorn *Android*, com ara interfícies o classes per realitzar funcions lògiques dins la capa de negoci ja depèn del propi *framework* utilitzat. Per aquest motiu recalco que es mostra el diagrama base que en definitiva és la branca que s'ha de tenir clara per al correcte desenvolupament del projecte.

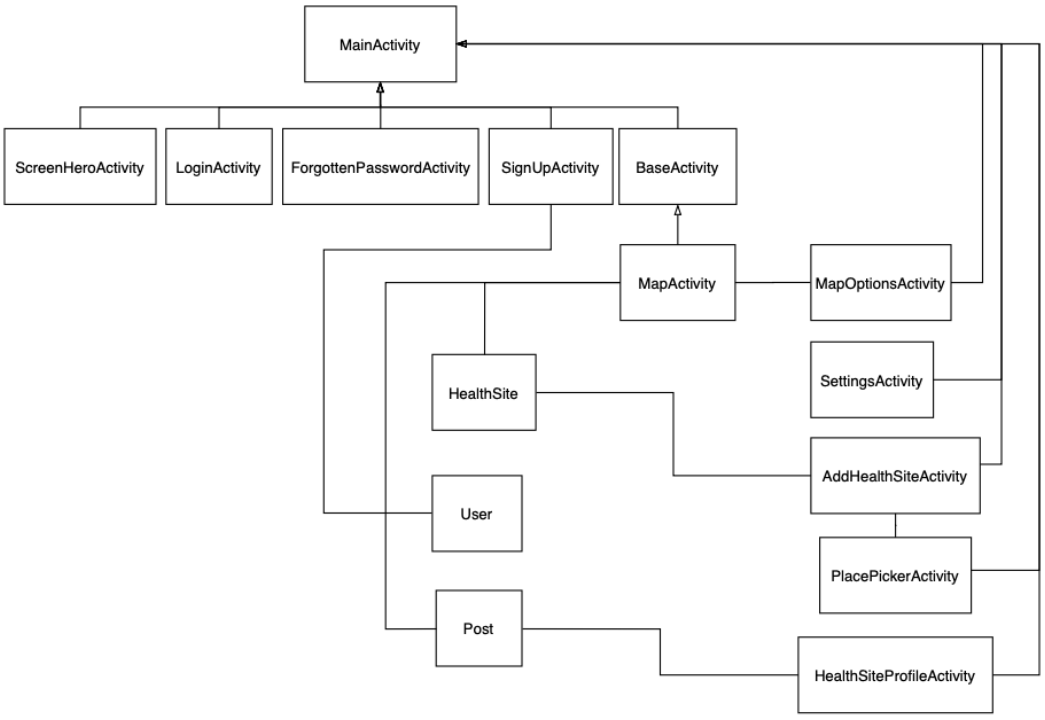


Figure 16: Diagrama UML principal

6.2.2 Arquitectura de la base de dades

La base de dades és l'encarregada de guardar les dades generades per l'aplicació per a la seva futura utilització.

S'utilitza *Firebase Firestore* que és una base de dades noSQL que emmagatzema les dades en format JSON. En la secció "3.3 Serveis utilitzats", setè paràgraf, es fa menció a la seva estructura més detalladament.

Una vegada es té clar el model del nostre sistema d'informació, només falta traslladar-lo a un model de dades comprensible per al nostre gestor de base de dades.

A continuació es mostrarà l'esquema corresponent a les diverses col·leccions i els seus documents. Però abans de tot, cal aclarir que al no seguir un esquema relacional, certes restriccions com les claus primàries no estan definides igual.

6.2.2.1 Col·lecció HealthSites

Encarregada d'emmagatzemar els documents corresponents a les "HealthSites", on cadascun s'identifica mitjançant un identificador únic i generat automàticament pel gestor a l'hora d'introduir un nou document.

```
HealthSites: {
    id: {
        address: string,
        adminArea: string,
        city: string,
        companyCIF: string,
        companyName: stirng,
        country: string,
        countryCode: string,
        description: string,
        ecologic: boolean,
        freeGluten: boolean,
        freeLactose: boolean,
        id: string,
        latitude: number,
        longitude: number,
        name: stirng,
        number: string,
        phoneNumber: string,
        position: {
            latitude: number,
            longitude: number
        },
        postalCode: string,
        restaurant: boolean,
        snippet: string,
        store: boolean,
        street: string,
        title: string,
        vegan: boolean,
        vegetarian: boolean,
        verified: boolean,
        webMail: string,
        webSite: string
    }
}
```

Figure 17: Schema HealthSites

6.2.2.2 Col·lecció Users

Aquí es guarden els documents amb les dades dels usuaris. S'utilitza també un identificador únic generat automàticament.

```
Users: {
    id: {
        dateOfBirth: number,
        email: string,
        gender: string,
        id: string,
        name: string,
        password: string,
        premium: boolean,
        surname: string,
        username: stirng
    }
}
```

Figure 18: Schema Users

6.2.2.3 Col·lecció Posts

En aquesta es guarden els Post o Comentaris realitzats pels usuaris. Cada document conté els camps propis del post, com ara el text corresponent i el *timestamp* de la publicació. Però s'hi guarden també l'*id* de la "HealthSite" a la qual s'ha comentat, i l'*id* de l'usuari que ha fet el comentari.

```
Posts: {
    id: {
        healthSiteId: string,
        id: string,
        text: string,
        timestamp: number,
        userId: string
    }
}
```

Figure 19: Schema Posts

6.2.2.4 Col·lecció Likes

La idea és tenir una sèrie de documents identificats per l'*id* del post corresponent, en els quals estiguin registrats tots els usuaris que han donat *like* al post esmentat.

Aquí, a diferència dels casos anteriors, l'identificador del document no es genera automàticament perquè no és viable tenir un document per cadascun dels *likes* realitzats.

Aleshores s'usa el mateix *id* del post, amb l'objectiu de relacionar aquest post amb tots els usuaris que l'hi hagin donat *like*.

```
Likes: {
    post_id: {
        user_id: boolean (true)
    }
}
```

Figure 20: Schema Likes

6.2.2.5 Col·lecció HealthSiteFollowers

Igualment com el cas anterior, tenim per identificadors els *ids* de les "HealthSites", i en els documents s'hi guarden els *ids* dels usuaris que les segueixen.

```
HealthSiteFollowers: {
    healthSite_id: {
        user_id: boolean (true)
    }
}
```

Figure 21: Schema HealthSiteFollowers

6.2.2.6 Col·lecció UserFollowing

Altrament, s'utilitza com identificadors els *ids* dels usuaris, i s'hi guarda els *ids* de les HealthSite a les quals l'usuari segueix.

Com es pot apreciar, aquesta col·lecció i l'anterior tenen molt en comú. Realment guarden les mateixes dades però de diferent perspectiva. Això s'ha fet per requeriment del codi implementat ja que per certes funcionalitats i a causa de les limitacions de l'api per a la gestió de la base de dades, doncs l'única manera era mantenir la consistència entre dues col·leccions.

Per exemple, si es vol saber quants seguidors té una "HealthSite", és molt més fàcil comptar els camps que conté el document específic de la col·lecció "HealthSiteFollowers", perquè no és viable per rendiment, ni possible segons l'api, realitzar una consulta per extreure de la col·lecció "UserFollowing" el nombre d'usuaris que segueixen una "HealthSite".

L'altre cas es troba amb la mateixa problemàtica però a l'invers. Saber quantes "HealthSites" segueix un usuari.

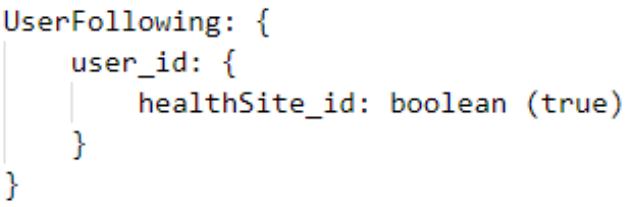


Figure 22: Schema UserFollowing

6.2.2.7 Col·lecció HealthSiteReport

En aquesta col·lecció, s'hi guarden els usuaris que han reportat una "HealthSite" en concret. D'aquesta manera, la idea radica en comptabilitzar quants usuaris i quins han reportat una "HealthSite".

Aleshores, si s'observa que una "HealthSite" té molts reports contínuament doncs es poden prendre mesures.

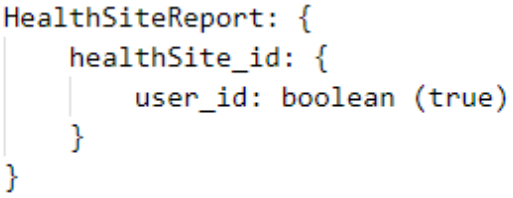


Figure 23: Schema HealthSiteReport

6.2.2.8 Col·lecció Feedback

La darrera col·lecció s'utilitza per guardar els missatges que els usuaris ens vulguin fer arribar per així tenir un canal obert on ells puguin expressar les seves idees, problemes que hagin trobat, possibles millores, qualsevol crítica constructiva, etc. Tot el *feedback* és benvingut.

Cada document té un *id* únic auto-generat, i conté un camp per identificar l'usuari, el propi text del missatge i el *timestamp* de l'instant en què es va realitzar.

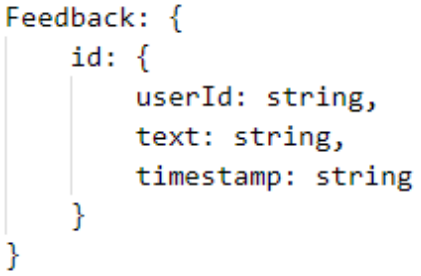


Figure 24: Schema Feedback

6.3 Prototip

El prototip és un primer concepte representat gràficament sobre allò que es vol desenvolupar; en el nostre cas una aplicació per a dispositius mòbils.

Ens dona una idea de com quedarà l'aplicació a desenvolupar, a nivell d'interfície i disseny, sense haver de començar a programar perquè es realitza en les fases inicials del projecte.

És important tenir clar què es vol fer, i com plasmar la idea abans de començar a picar codi sense tenir un rumb clar. Si no, malauradament tocarà refactoritzar i refer codi per implementar o retocar aquelles funcionalitats que es van codificar per intuïció. En conseqüència, pèrdua de temps i de diners en el cas d'estar en un entorn professional.

Tampoc s'ha de deixar l'aplicació dissenyada i amb l'aparença final. Simplement, serveix per estructurar com es vol organitzar les funcionalitats dins les pantalles, i què funciona i què no en el conjunt de l'aplicació.

6.3.1 Esbós inicial/Wireframe

L'esbós inicial o *wireframe* defineixen, sense res d'estil o disseny, quines coses han d'aparèixer a les pantalles i a on. El *wireframe* constitueix l'esquelet de l'aplicació mòbil.

Es va decidir optar per realitzar un *wireframe* per sobre d'altres alternatives com ara els *mockups* perquè es volia separar el disseny visual en les primeres fases, fins a tenir clar com es volia organitzar tot plegat.

En les següents imatges es presenta d'una forma senzilla i esquemàtica com es visualitzava aleshores la idea inicial.

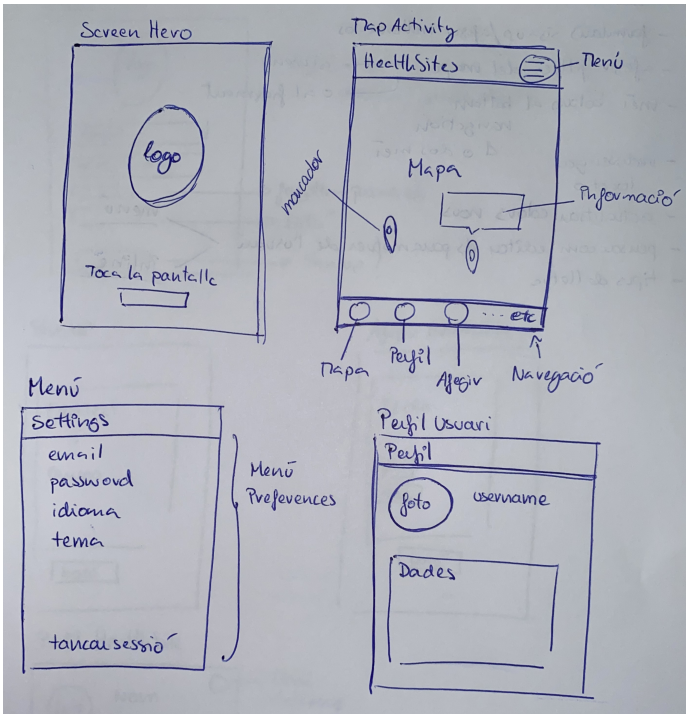


Figure 25: Esbós 1

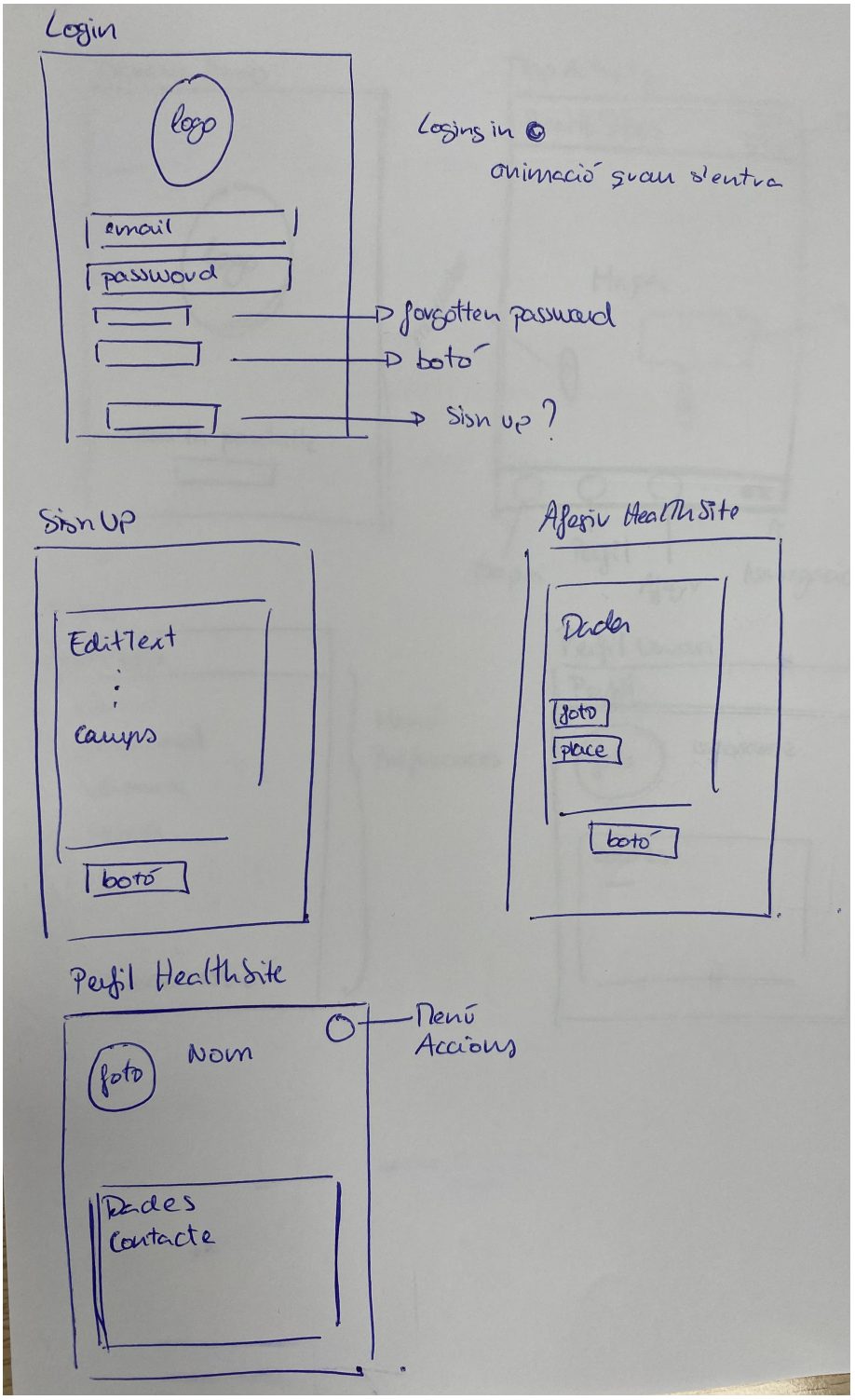


Figure 26: Esbós 2

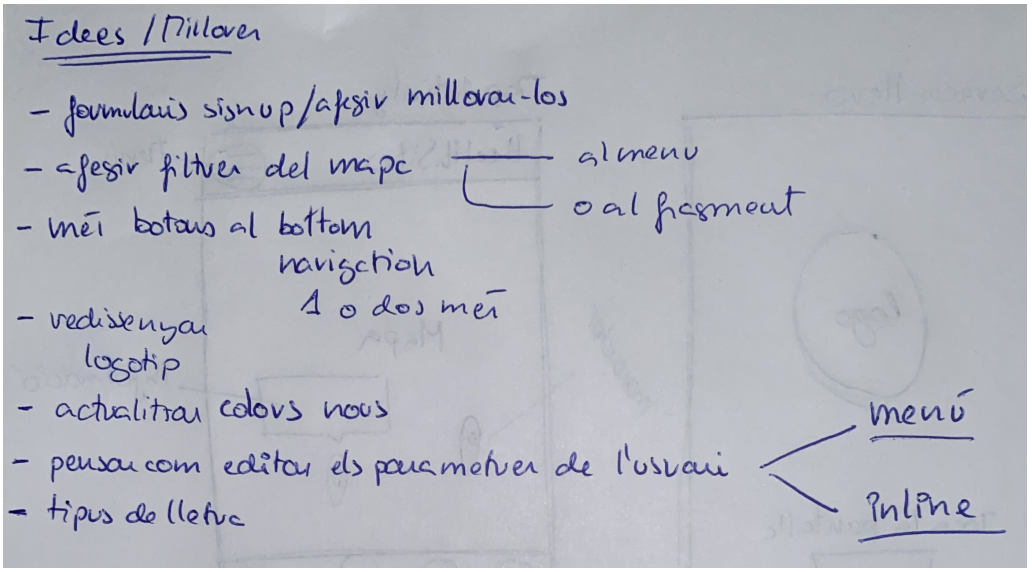


Figure 27: Idees de millora inicials

6.4 Disseny de la interfície

La nova versió de la interfície d'usuari es va dissenyar conjuntament amb l'ajuda d'una dissenyadora.

Es volia donar una nova imatge, un nou estil i redissenyar el logotip. Es va començar per establir una paleta de colors, els quals formen el nou logotip de l'aplicació.

Figure 28: Nou logotip

Després, vam abraçar el disseny de les funcionalitats principals. El *login*, els formularis, la pantalla del mapa, etc.

A mesura que, tant ella com jo o a partir del *feedback* rebut a través de les proves d'usabilitat trobàvem idees noves, ens posàvem en contacte per consultar els possibles canvis i decidir si tirar-ho endavant. Tot això amb l'objectiu d'aconseguir una interfície entenedora, amigable, i centrada en l'usuari aplicant el DCU.

També quan es realitzava les funcionalitats marcades s'acordava una reunió per parlar sobre possibles millores i validar la interfície. En relació amb la metodologia utilitzada, es decidia aleshores si incrementar o abans realitzar alguns retocs/millores.

És cert que des del principi es va decidir l'estil base, però no va ser fins a les últimes converses on es va acabar de polir totes les petites parts, icones, animacions entre pantalles, etc.

En resum, va ser de gran aprenentatge perquè tot i tenir les idees clares a mesura que s'anava avançant i presentant noves funcionalitats implementades ja validades, tot començava a prendre forma. Va ser un camí realment.

A continuació es mostrarà el corresponent mapa de navegació entre les pantalles i seguidament un recull de com ha quedat el disseny i estil final d'aquestes mateixes amb unes breus explicacions sobre la funcionalitat de cadascuna.

6.4.1 Mapa de navegació entre pantalles

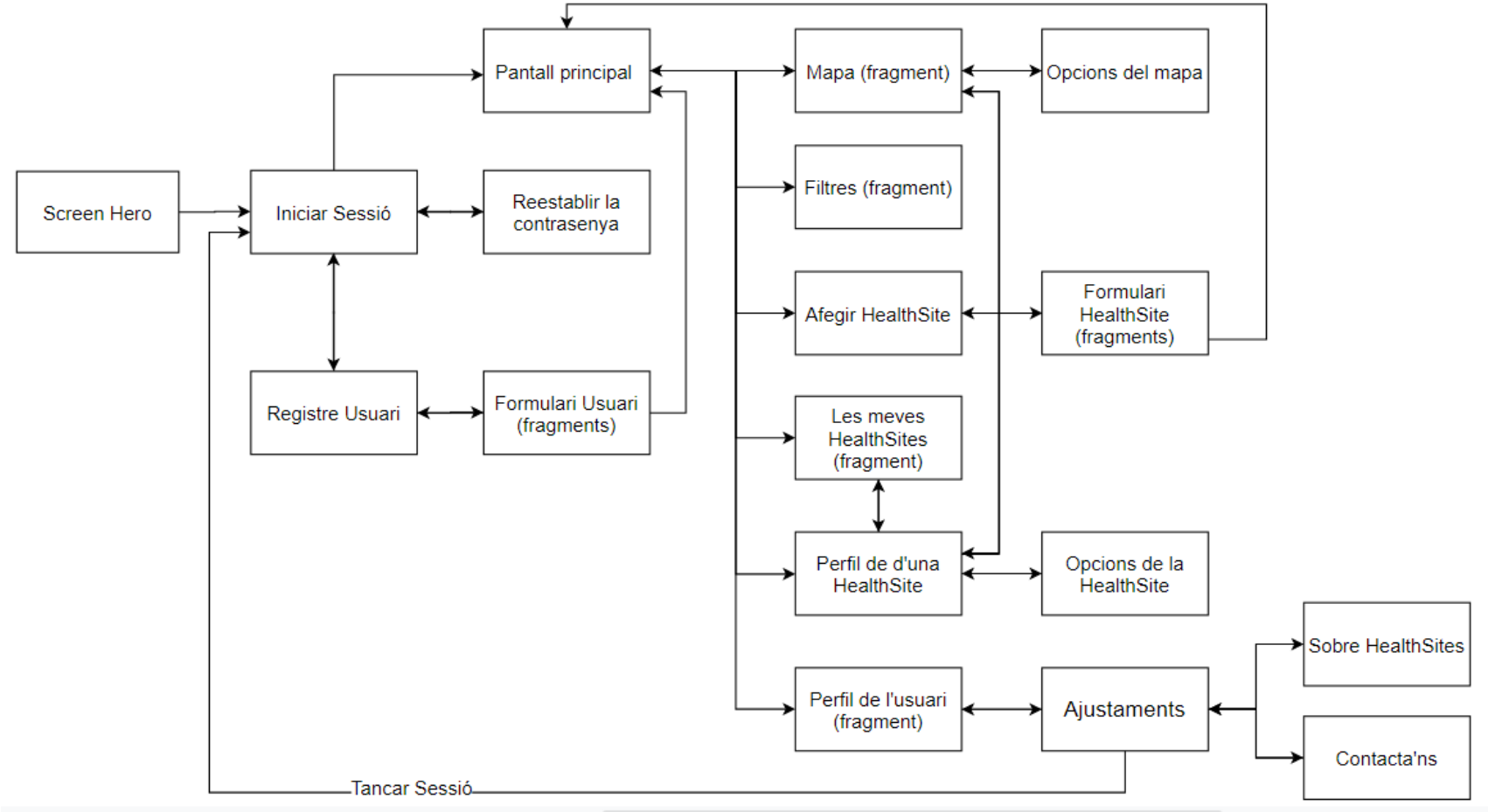


Figure 29: Diagrama de navegació entre les pantalles

6.4.2 Disseny de les pantalles

6.4.2.1 Screen hero i Inici de sessió

Són les dues pantalles inicials de l'aplicació. En la figura 29 es pot observar la portada de l'aplicació on és dona especial pes al logotip de la mateixa. Per l'altre costat, en la figura 30, la següent pantalla, es pot veure la pantalla per iniciar sessió.

Figure 30: Pantalla Screen hero

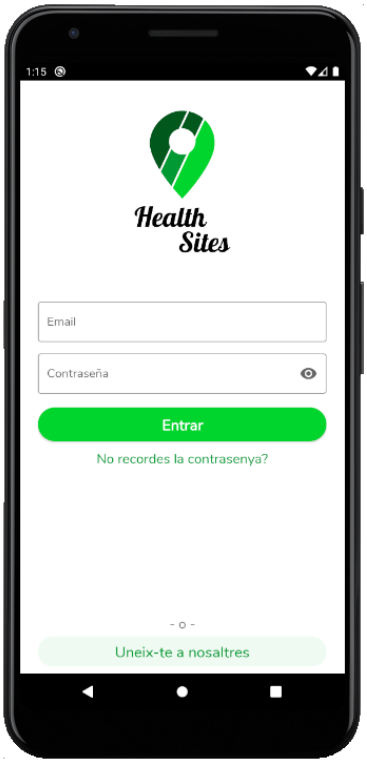


Figure 31: Pantalla Inici de Sessió

6.4.2.2 No recordes la contrasenya?

Aquesta es una pantalla de suport per aquells usuaris que hagin perdut o oblidat la contrasenya. Aleshores, introduint el seu correu electrònic podran re-establir la seva contrasenya gràcies al correu que el sistema els hi enviarà. Sempre que sigui un correu vàlid.

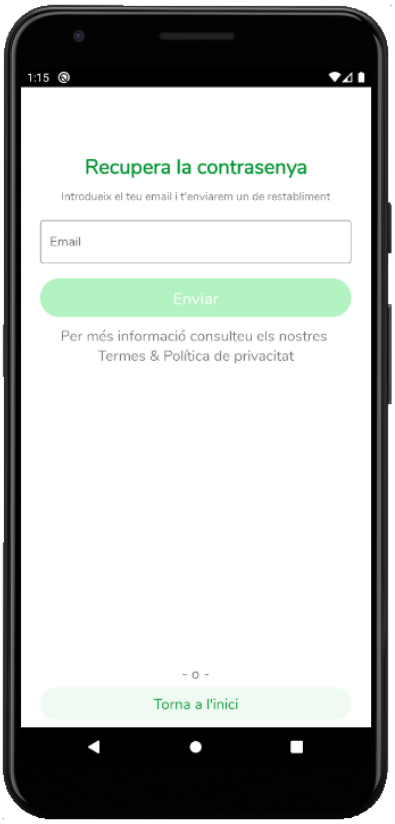
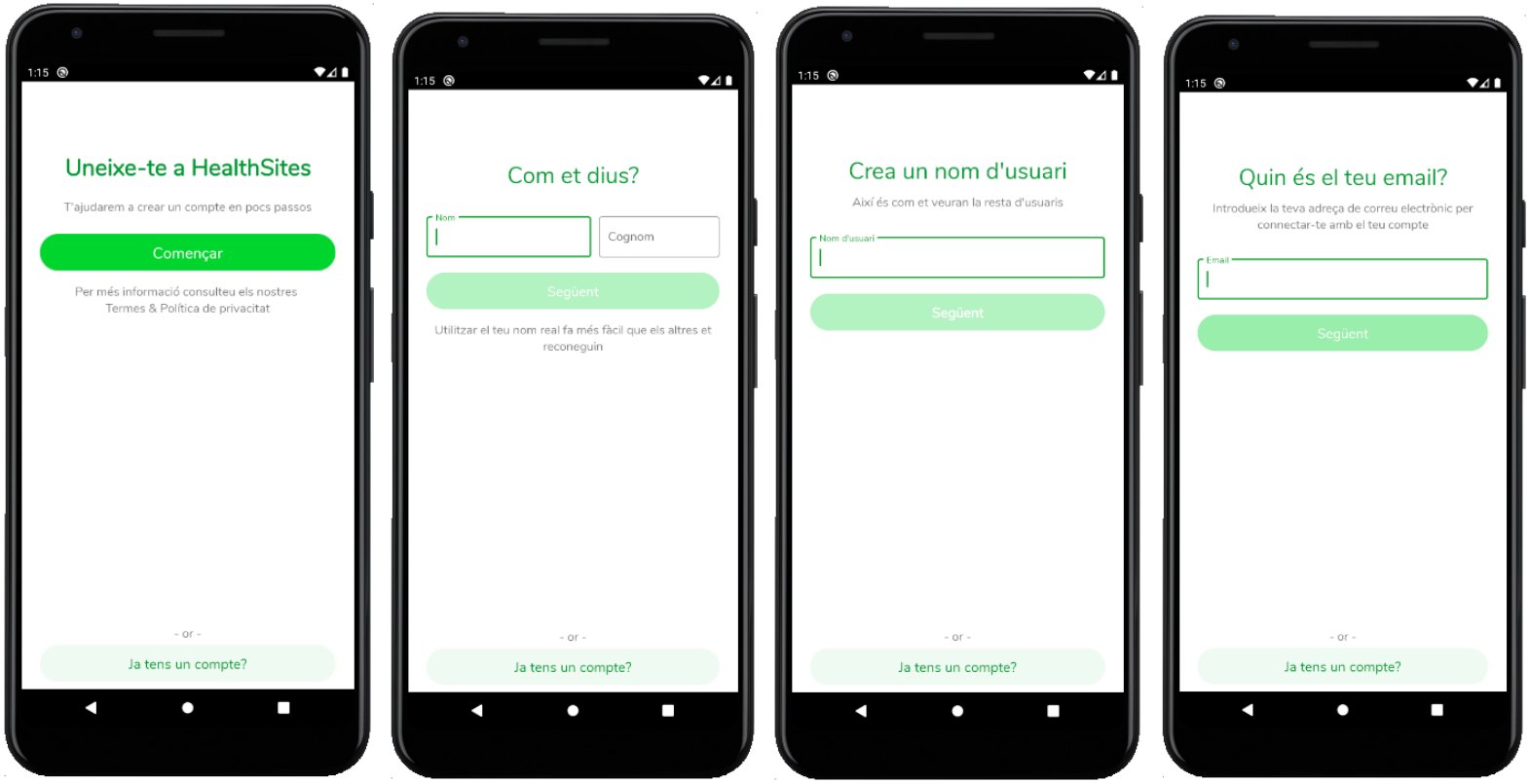


Figure 32: Pantalla de recuperació de la contrasenya

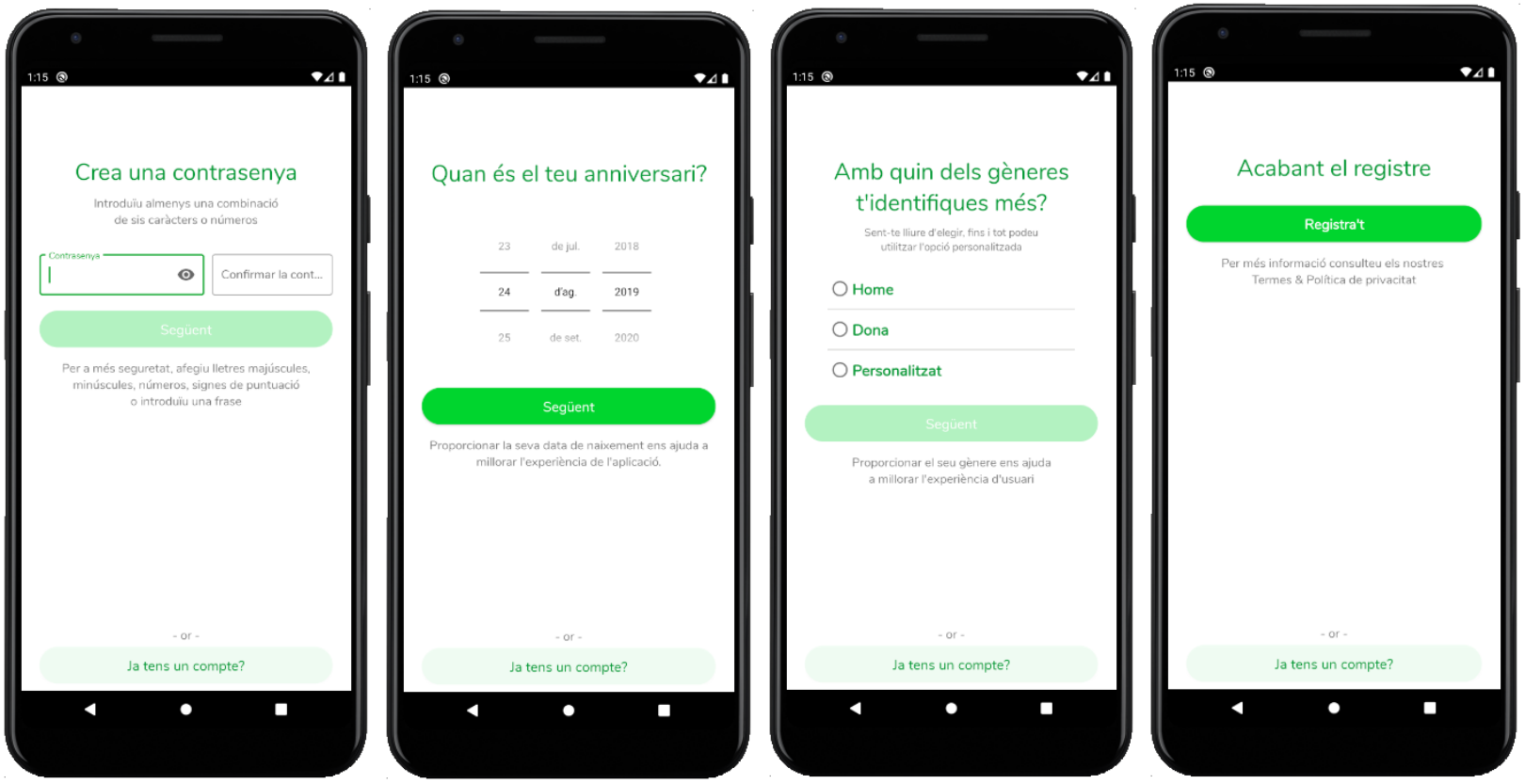
6.4.2.3 Formulari d'inscripció

Seguidament trobareu un recull de cadascuna de les subpantalles què formen el formulari d'inscripció per a nous usuaris. Més endavant en la secció "Implementació" es dedica un punt per explicar com s'ha codificat tot plegat utilitzant fragments.



(a) Començar (b) Nom i cognom (c) Nom d'usuari (d) Email

Figure 33: Formulari inscripció part 1



(a) Contrasenya (b) Anniversari (c) Gènere (d) Acabant el registre

Figure 34: Formulari inscripció part 2

6.4.2.4 Mapa

Aquí s'observa en la figura 35 la pantalla principal de l'interior de l'aplicació, el mapa. En ell podeu veure la ubicació de l'usuari i les diferents "HealthSites" que té al voltant.

A més a més, en la figura 36 s'hi mostra el contingut del menú situat a la part superior dreta de la pantalla del mapa. El qual, correspon a les diferents opcions de configuració i personalització del mateix.

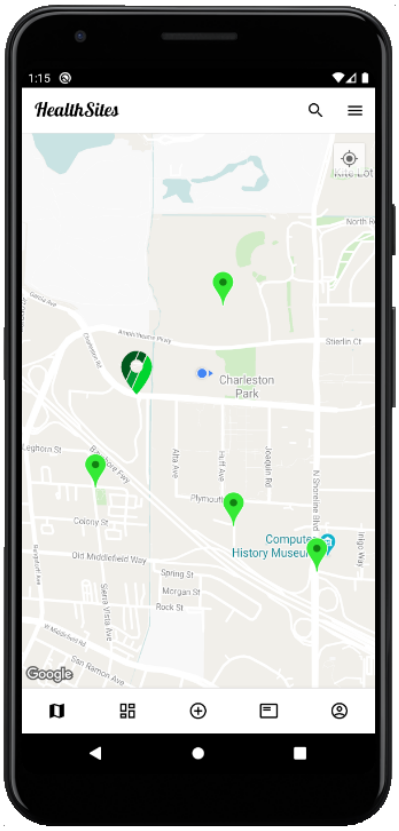
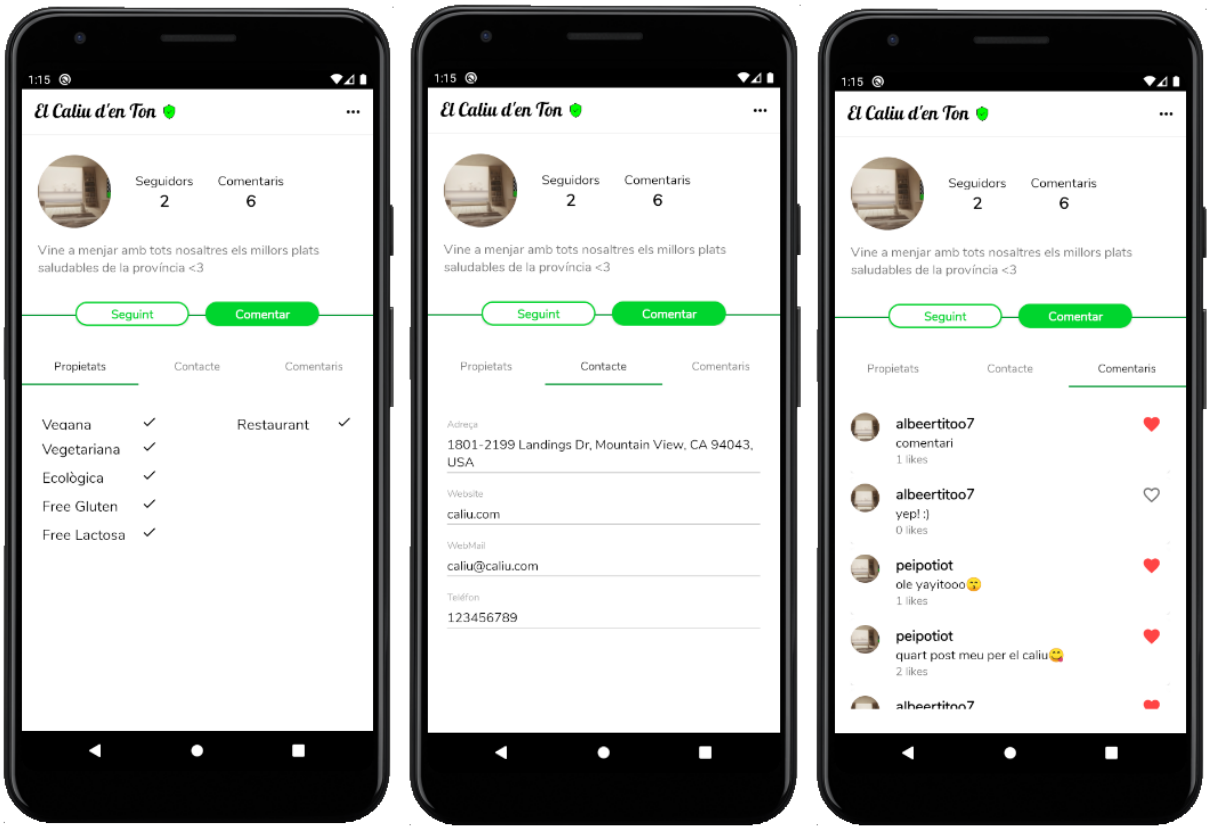


Figure 35: Pantalla del Mapa

Figure 36: Menú d'opcions del mapa

6.4.2.5 Perfil de la HealthSite

Aquest seria l'estil final del perfil d'una "HealthSite". S'hi mostren tres imatges corresponents a les seccions "Propietats", "Contacte" i "Comentaris" respectivament.

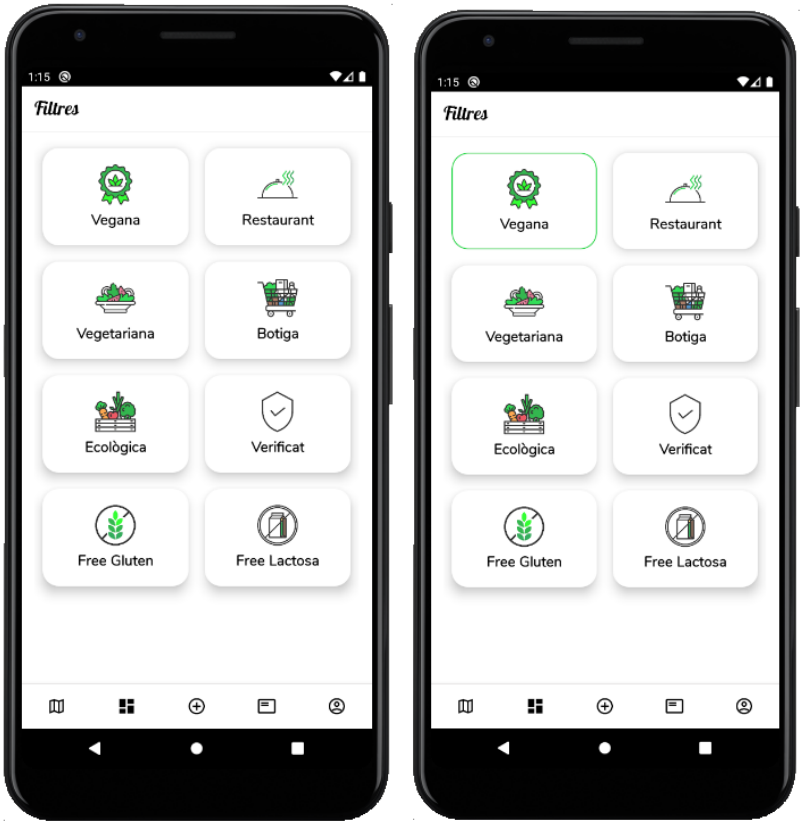


(a) Propietats (b) Contacte (c) Comentaris

Figure 37: Pantalles del perfil d'una HealthSite

6.4.2.6 Filtres

En les següents imatges es mostra la pantalla de filtres, on pots seleccionar les propietats de les "HealthSites" que desitges que se't mostrin al mapa. Aleshores, quan es torni al mapa es filtraran els marcadors.



(a) (b) Filtre seleccionat

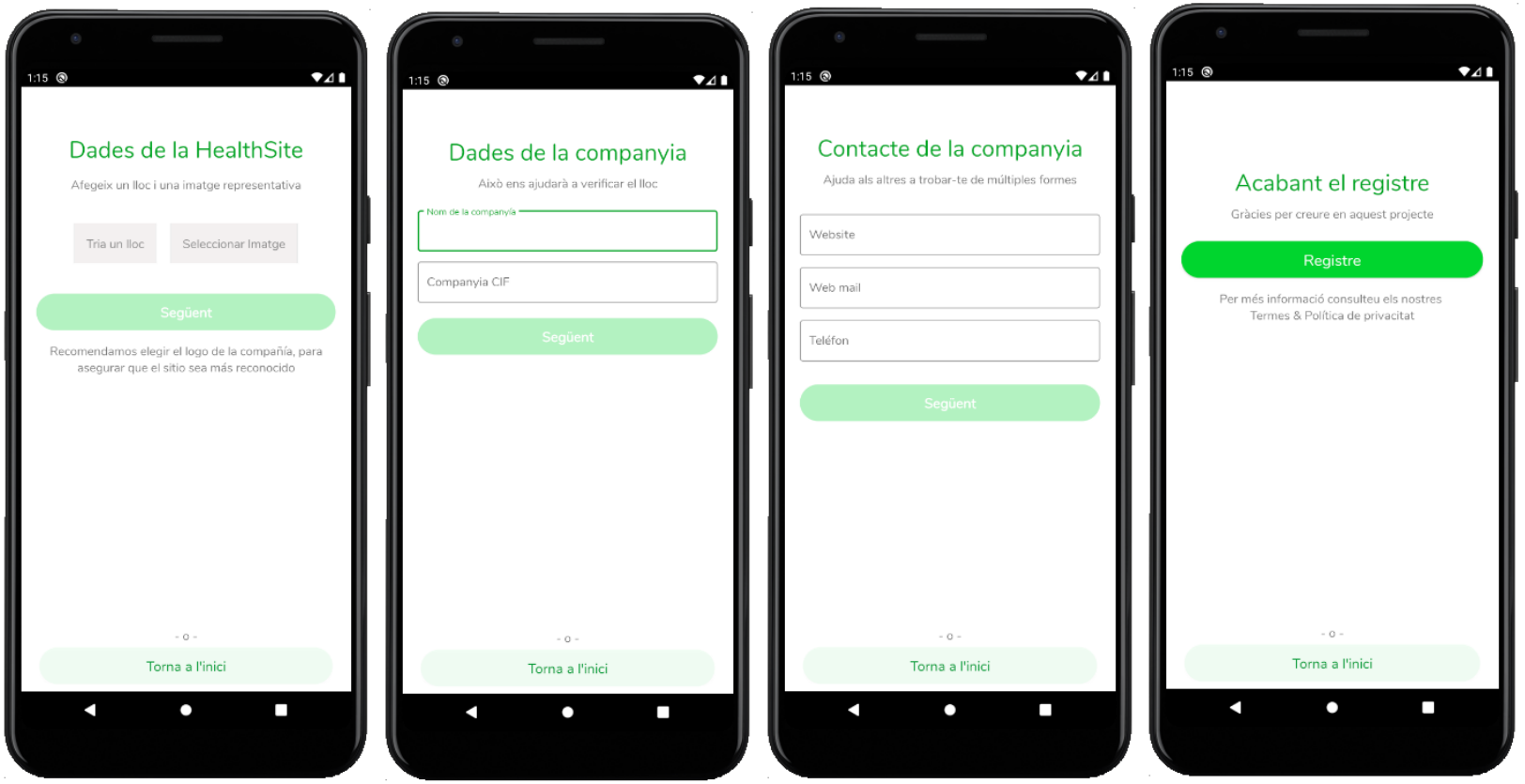
Figure 38: Pantalla dels filtres

6.4.2.7 Formulari de registre d'una HealthSite

Seguidament es recullen les imatges respectives al formulari, seguint el mateix estil marcat pel d'inscripció.

(a) Començar (b) Dades (c) Propietats (d) Definició

Figure 39: Formulari registre d'una HealthSite part 1



(a) Localització i fotografia (b) Dades de la companyia (c) Contacte de la companyia (d) Acabant el registre

Figure 40: Formulari registre d'una HealthSite part 2

6.4.2.8 Les meves HealthSites

En aquesta pantalla s'hi mostra una llista de les "HealthSites" que l'usuari segueix. Des d'aquí pot realitzar certes accions de forma ràpida, com ara comentar o deixar de seguir; inclús pot saber fàcilment el número de seguidors i comentaris. Encara que, inicialment es va pensar per a què només fos un accés ràpid al perfil de cadascuna.

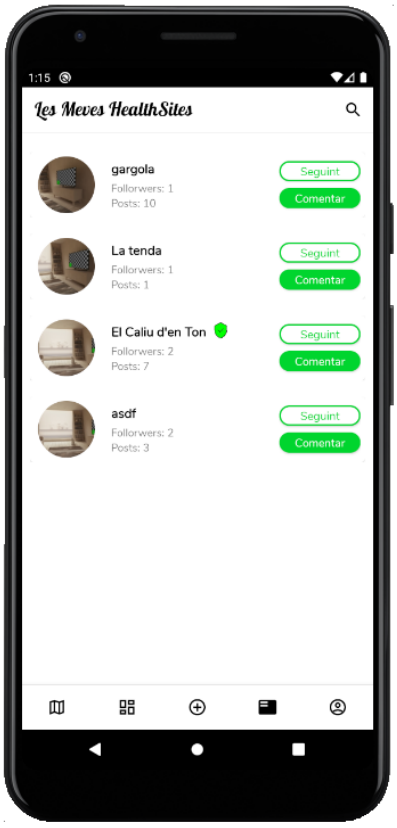


Figure 41: Pantalla de les meves HealthSites

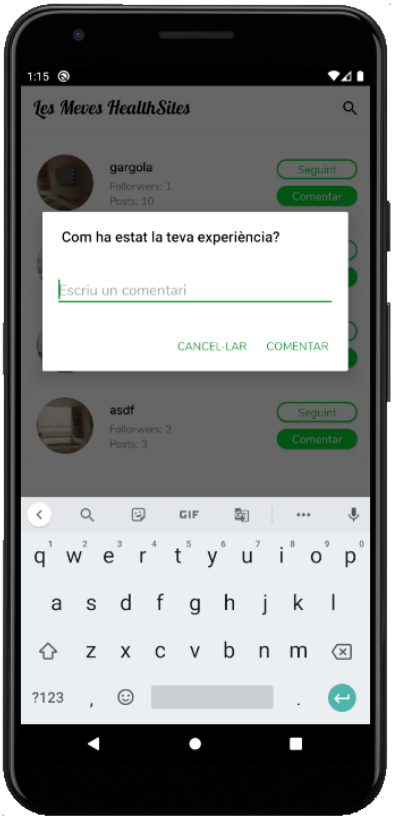


Figure 42: Comentar a la pantalla de les meves HealthSites

6.4.2.9 Perfil de l'usuari i Ajustaments

En la figura 43 es presenta el disseny final del perfil de l'usuari en el qual es podrà editar els diferents camps i la fotografia, com saber en un cop d'ull el número de "HealthSites" seguides i quants comentaris s'han realitzat.

D'altra banda, es podrà accedir des del botó de la part superior dreta al menú d'ajustaments del sistema on s'hi trobarà diferents opcions des de configurar l'idioma, canviar les credencials... fins un apartat de contacte per rebre *feedback* dels usuaris.

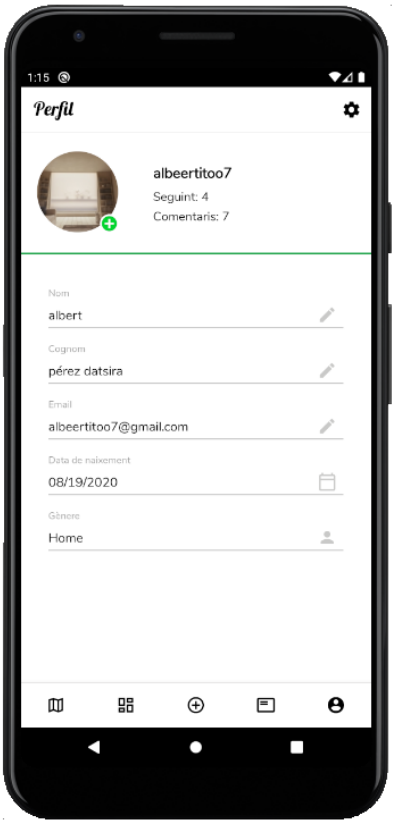


Figure 43: Pantalla del perfil de l'usuari

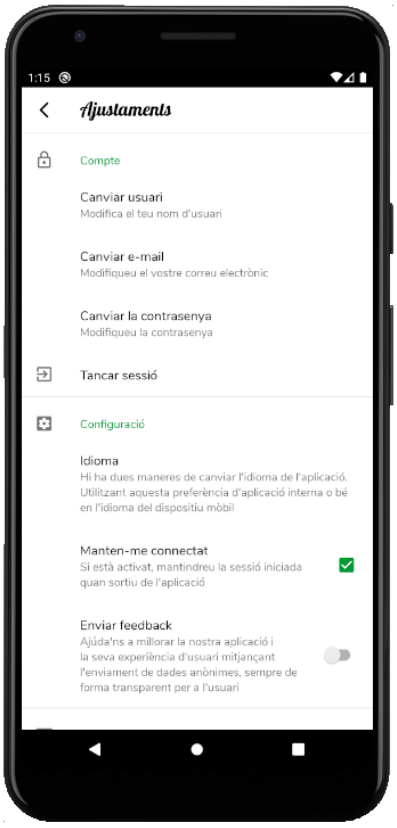


Figure 44: Pantalla dels ajustaments

6.4.2.10 Contacta'ns

Als ajustaments hi ha una opció del menú dedicada per a què els usuaris puguin contactar fàcilment i deixar comentaris sobre el que creguin més convenient.

Figure 45: Pantalla per contactar

6.4.3 Disseny de menús

Es buscava trobar un estil fàcil de llegir i entenedor. L'estil per defecte del menú de preferències semblava molt correcte, i es va decidir aplicar la seva estructuració. Però donant-li algun toc especial, com ara afegir certes icones relacionades amb el significat, canviar la lletra tant del text com del títol, dividir el menú en diferents categories, i afegir resums en cadascuna de les opcions del menú per deixar clar la seva funció.

A més, aquest estil de menú proporciona consistència dins de la plataforma *Android*, doncs és el què s'utilitza de manera estàndard.

En la figura 44 trobareu un exemple.

6.4.4 Estil de navegació

Per estil de navegació es poden entendre diferents conceptes. Si parlem de les animacions entre pantalles, es va triar aplicar una forma senzilla i no enrevessada. Les típiques "*slide*" i "*fade*". Tant "lliscar cap a dins" (*slide-in*) com "lliscar cap a fora" (*slide-out*), i també les "fer se visible" (*fade-in*) i desaparèixer (fade-out).

Per exemple, quan ha d'aparèixer un menú o canviar entre fragments d'un formulari, s'utilitza un "lliscar cap a dins des de la dreta" i la pantalla actual se'n va usant un "lliscar cap a l'esquerra".

Però en alguns casos també s'utilitza un "fer-se visible" o "desaparèixer" com en els casos dels submenús.

Són animacions ràpides i que no cansen a l'usuari, a més a més donen dinamisme a la interfície.

Per l'altra part, tenim l'anomenada "*bottom navigation*", una barra situada a la part inferior de la pantalla principal de l'aplicació la qual et permet navegar entre diversos fragments/funcionalitats i activitats, on cadascuna d'aquestes navegacions també estan animades.

Els botons et porten, començant d'esquerra a dreta, al mapa, als filtres, al formulari per afegir una "HealthSite", a la pantalla de les "HealthSites" que segueix l'usuari, i finalment al perfil.

Figure 46: Bottom navigation

6.4.4.1 Bottom Navigation

Aquesta tipus de navegació dóna bons resultats gràcies a la seva usabilitat i enteniment perquè actualment és molt popular entre les aplicacions disponibles en el mercat i ràpidament són accessibles totes les seccions de l'aplicació.

Les icones s'han triat específicament per a ser intuïtives i clares, i en el cas de no saber què significa alguna, el contingut de la seva pantalla és fàcil de comprendre. A part, s'han posat *dialogs*, pop-ups en format de missatge, d'informació per quan l'usuari entri per primera vegada a les respectives pantalles.

Per altra banda, es volia donar importància al botó per afegir "HealthSites" centrant-lo a la barra de navegació perquè una part important de la futura campanya de màrqueting o pla de negoci seria promoure la construcció de la base de dades. Aleshores es va decidir que per incloure aquest botó totalment centrat es necessitaven un nombre imparell d'opcions. És per això, i també per les recomanacions de la documentació d'*Android* la qual esmenta que és millor afegir 5 o 3 opcions que 4, doncs es va decidir apostar per aquesta disposició.

6.4.5 Formularis

En la primera versió de l'aplicació es van usar formularis estàtics d'una sola pàgina. Ja en les primeres fases del disseny, on es van començar a formular idees, es tenia clar les ganes de donar-li un nou aire als formularis.

Després d'investigar es va concloure que seria interessant realitzar formularis dinàmics per passos, on en cadascun dels anomenats "Steps" es vagi guiant a l'usuari a mesura que aquest introdueix les dades. Simplement per ser més entenedors i clars, i donar-li una experiència agradable i d'importància per a nosaltres. Podeu trobar les respectives pantalles a les figures 33, 34, 39 i 40.

Per concloure, es va apostar pels "fragments", definits més endavant en la subsecció "Fragments" de la secció "Desenvolupament de l'aplicació", els quals formen cadascuna de les parts del formulari respectiu.

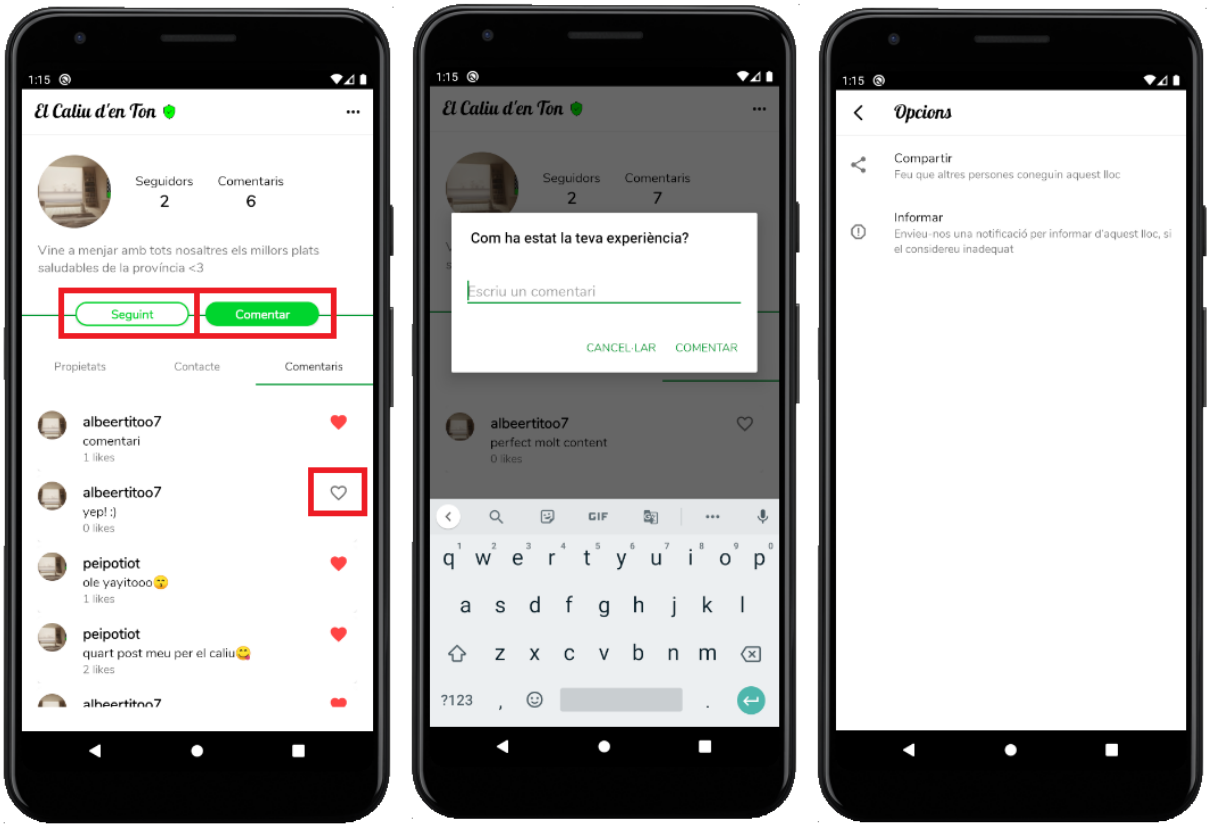
D'aquesta manera es dividia el formulari en parts o fragments, i quan se'n completava una el fragment visible marxava i entrava el pròxim tot usant animacions de transició.

6.4.6 Dirigir la interacció entre els usuaris i les HealthSites

En l'anàlisi de la competència es va trobar certes aplicacions que encaraven l'aplicació com una xarxa social d'interacció entre usuaris. No es volia donar aquesta sensació. Per tant tot està centrat a donar *feedback* de l'usuari cap a la "HealthSite".

L'usuari podrà donar *like* als comentaris, comentar sobre la seva experiència i seguir les més interessants per tenir un fàcil accés. A més a més dispondrà d'un menú d'opcions d'accés ràpid a la part superior dreta que li permetrà compartir la "HealthSite" o informar en el cas de creure què és inadequada.

I pel que fa al perfil de l'usuari únicament serveix perquè pugui accedir a les seves dades i modificar-les, i al menú d'ajustaments.



(a) Accions per interactuar (b) Afegir un comentari (c) Menú d'opcions

Figure 47: Interacció de l'usuari sobre una HealthSite

6.4.7 Lletra utilitzada

Juntament amb la dissenyadora vàrem buscar una lletra adient que no fos la típica que ve per defecte. Es volia canviar, i donar-li un toc personal, no tan rígid, i de caràcter alegre. Pel text normal s'ha posat la "Nunito", una lletra una mica més arrodonida però sense deixar de ser comprensible. I pel que fa als títols de l'*Action bar* s'ha posat l'anomenada "Pattaya" perquè volíem donar-li un aspecte especial i diferenciador, per remarcar l'estil propi de l'aplicació. Les dues sota la llicència de *Open Font License*.

Es van utilitzar els fitxers .ttf corresponents i es van incorporar al directori "/fonts" del projecte *Android*. Aleshores per aplicar utilitzant el fitxer "styles.xml" es va aplicar l'estil de lletra corresponent al elements.

```
<style name="Theme.MyApp" parent="Theme.MaterialComponents.Light.NoActionBar.Bridge">
    <item name="colorPrimary">@color/colorGreenDark</item>
    <item name="colorPrimaryDark">@android:color/black</item>
    <item name="android:textColorPrimary">@android:color/black</item>
    <item name="android:windowBackground">@color/White</item>
    <item name="colorAccent">@color/GreenHealth</item>
    <item name="fontFamily">@font/nunito</item>
    <item name="android:toolbarStyle">@font/pattaya</item>
</style>
```

Figure 48: Configuració de les fonts en el fitxer styles.xml del projecte

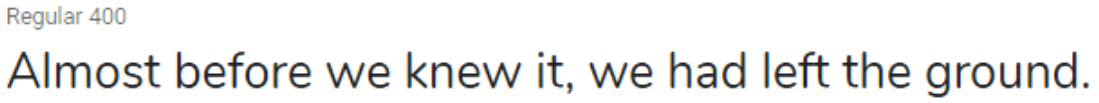


Figure 49: Font Nunito

Figure 50: Font Pattaya

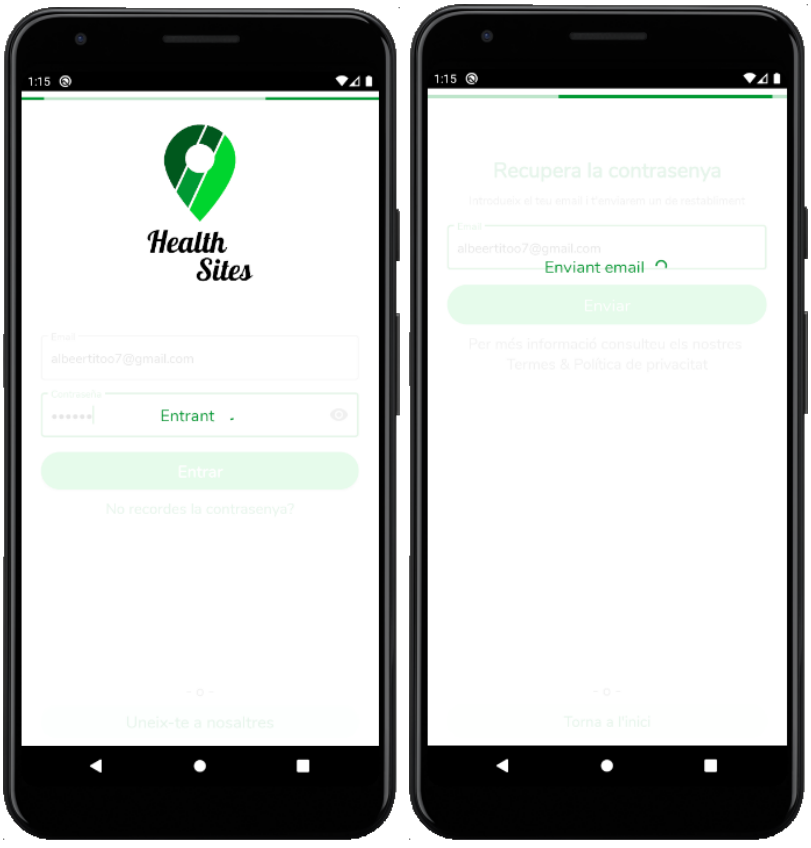
6.4.8 Animacions d'espera

Les animacions d'espera són aquelles que col·laboren en donar *feedback* a l'usuari sobre què s'està treballant per completar una acció concreta.

Per exemple, quan s'ha d'iniciar sessió el procés no és instantani i aleshores s'utilitza una petita animació per fer-li saber a l'usuari que s'està treballant.

Les pròximes imatges mostren dos exemples d'animacions. El primer quan es vol iniciar sessió i el segon quan es vol recuperar la contrasenya. Es volia donar la informació precisa a l'usuari però utilitzant un estil minimalista sense fer aparèixer grans "loadings" ni massa text.

Com es pot observar en la figura 49, les animacions es componen per dues "ContentLoadingProgessBar" i un petit text centra just al costat esquerra d'una d'elles. Les dues "ProgressBar" són personalitzades per obtenir aquestes formes, la superior en format barra i la del costat del text en format circular.



(a) Animació d'iniciar sessió (b) Animació de recuperar la contrasenya

Figure 51: Mostra d'animacions d'espera

6.4.9 Icones i metàfores

Seguidament trobareu una taula que llista les principals icones de l'aplicació juntament amb el seu significat.

	Mapa
	Filtres
	Afegir "HealthSite"
	Les meves "HealthSites"
	Perfil de l'usuari
	Ajustaments
	Menú
	Menú d'opcions
	Mostrar la contrasenya
	Buscador
	Donar *like*
	Marcador "HealthSite" verificada
	Marcador "HealthSite" no verificada
	Verificat
	Enrere
	Editar
	La "HealthSite" disposa de la propietat
	Afegir fotografia al perfil d'usuari

Table 7: Descripció de les icones principals

6.5 Proves d'usabilitat

Quan s'arribava al final d'una iteració, la nova versió operativa conjuntament amb la seva interfície s'havia de testejar.

Amb l'ajuda de la dissenyadora es provaven les funcionalitats i les noves millores incorporades, i tot estar molt limitat per la pandèmia viscuda per trobar usuaris per realitzar les proves es va aconseguir dues persones externes al projecte disposades a testejar la interfície (ma mare i el meu germà petit).

Donat que les proves amb els dos voluntaris s'han repetit en 4 ocasions, per tant s'ha tingut en compte el *feedback* de l'usuari final, podríem dir que s'ha aplicat, en certa mesura, la metodologia del Disseny Centrat en l'Usuari (DCU). Entenent que l'usuari és aquell que ha de comprendre la interfície i la funcionalitat, perquè tot *feedback* que es rebi dóna un futur a l'aplicació.

A continuació trobareu un recull d'imatges on s'aprecien diferents notes extretes de les validacions que van fer els voluntaris en diferents moments del procés de desenvolupament del software que van servir per millorar els punts que des de la perspectiva de desenvolupador no es consideren:

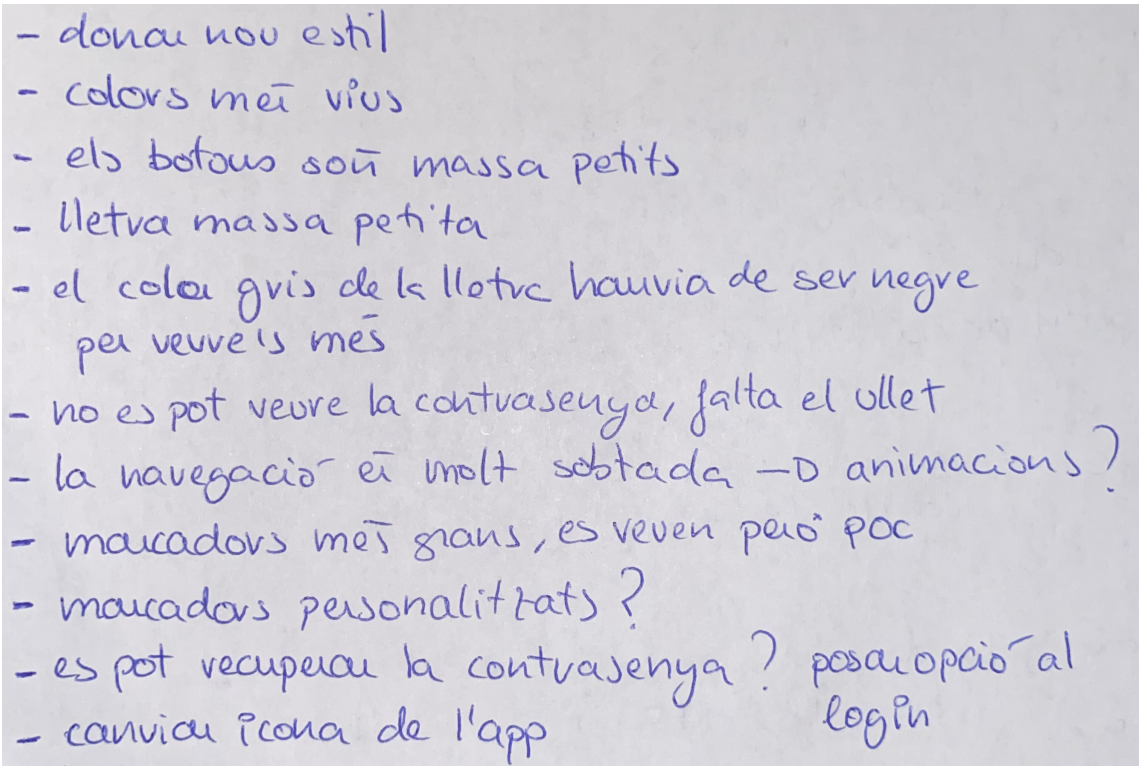


Figure 52: Notes de les proves d'usabilitat 1

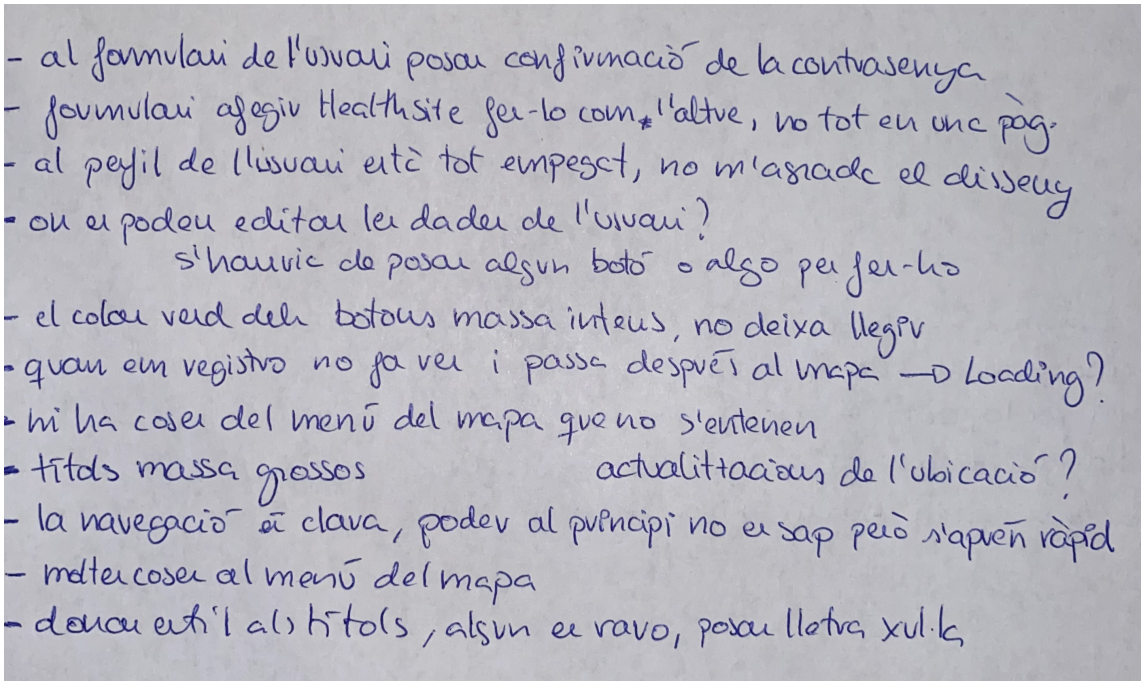


Figure 53: Notes de les proves d'usabilitat 2

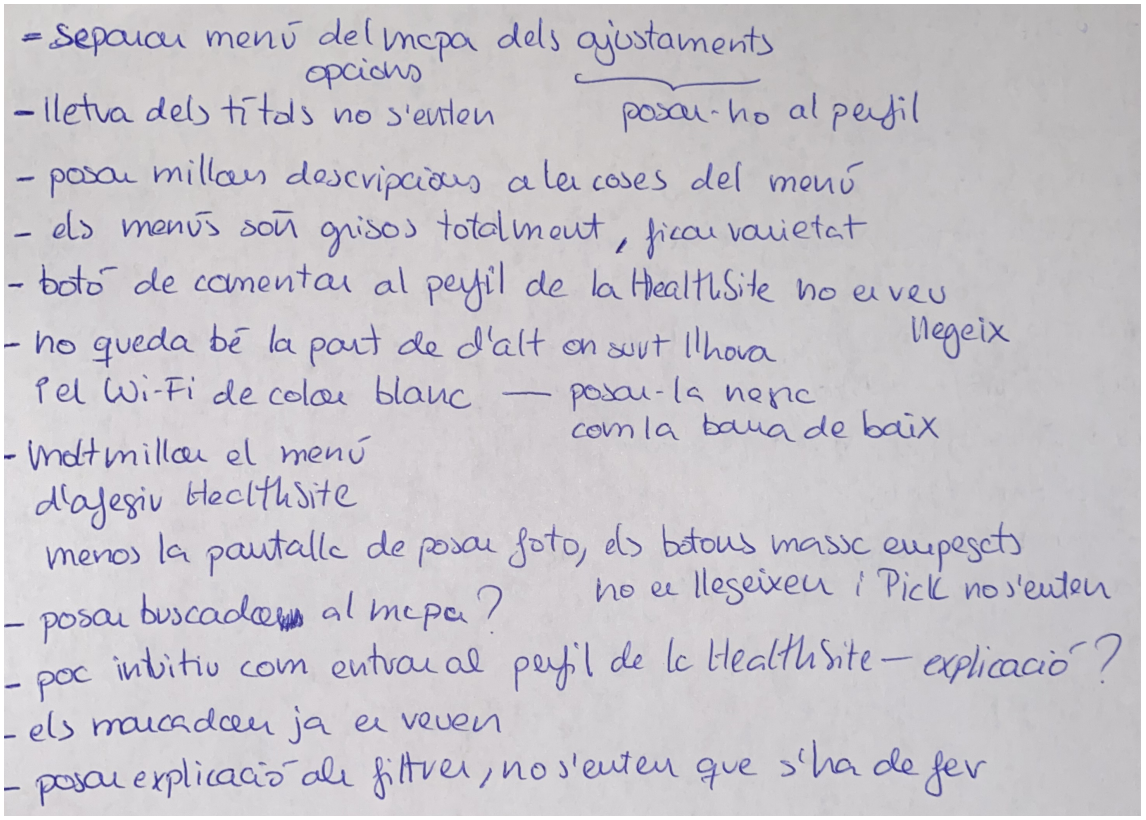


Figure 54: Notes de les proves d'usabilitat 3

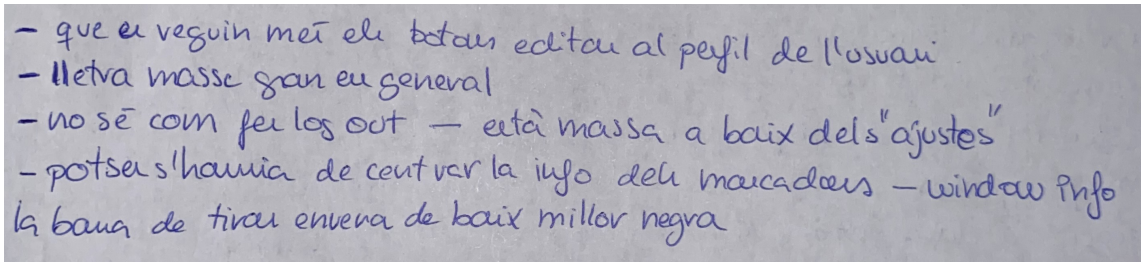


Figure 55: Notes de les proves d'usabilitat 4

7 Implementació

7.1 Decisions d'implementació

7.1.1 De RealTime Database a Firebase Firestore

RealTime Database i *Firebase Firestore* són dues bases de dades del mateix *Firebase*, però tot i tenir el mateix format per guardar les dades (JSON), tenen algunes petites diferències tant en les consultes que s'hi poden fer com en l'estructuració de la jerarquia. En la subsecció "Serveis utilitzats" de la secció "Descripció de la tecnologia utilitzada" s'explica detalladament les seves diferències.

Inicialment a l'aplicació es va optar per utilitzar la base de dades *RealTime Database*. Però, a causa del fet que no permetia pujar imatges i tampoc estava preparada per a una escalabilitat gran, es va decidir canviar a *Firebase Firestore* per guardar les dades i *Cloud Storage*, un altre servei de *Firebase*, per guardar les imatges conjuntament.

Va ser una decisió correcta, tot i haver de fer una reestructuració del codi i reescriure totes les parts que es comunicaven amb la base de dades. En conseqüència, es va haver de realitzar un altre cop totes les proves de funcionament i consistència del nou codi escrit, i assegurar-se del correcte funcionament.

7.1.2 Llista de Posts/HealthSites pròximes

Es tenia pensat afegir una funcionalitat a l'aplicació. Consistia en poder llistar, ja fossin les "HealthSites" més properes a la ubicació de l'usuari o els últims comentaris de les "HealthSites" més properes a l'usuari. Però, en usar un servei extern com *Firebase*, s'estableix una dependència. En conseqüència la seva API per gestionar la base de dades no permetia realitzar consultes compostes sobre camps diferents, ja que la idea era triangular la posició utilitzant l'objecte compost "LatLong" (Longitud i Latitud). Finalment, la idea va quedar en l'aire a causa de no poder-la portar a la pràctica.

7.1.3 De Java a Kotlin

La compatibilitat entre aquests dos llenguatges ha estat clau en el desenvolupament. S'ha de dir que la documentació de *Kotlin* ha millorat moltíssim durant els darrers anys, almenys l'oficial. Però en el projecte s'han utilitzat certes classes o llibreries externes de gent que treballa pel seu compte i ho comparteix.

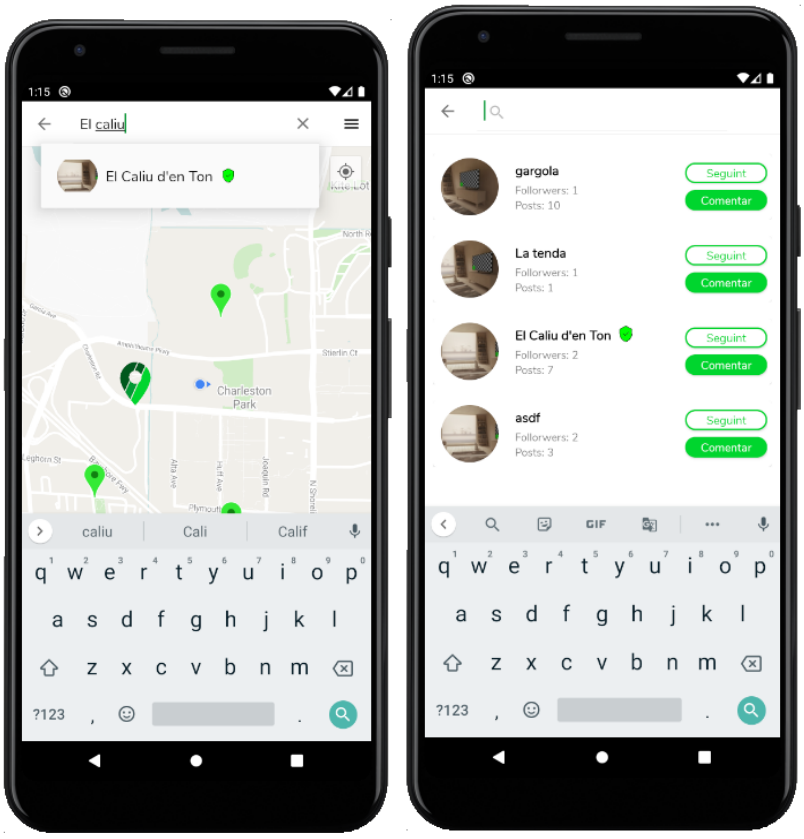
L'entorn et permet passar fitxers de Java a *Kotlin*, tot i que de vegades no ho fa del tot correctament. Però ha estat una eina molt útil en aquests casos perquè et facilita la feina a l'hora de convertir classes i fitxers.

Altrament, quan hi havia certes parts del codi a implementar amb *Kotlin* que no es trobava la corresponent documentació o s'estava encallat, es va realitzar la codificació en Java i es va passar a *Kotlin*, després de testejar-ho, i es va incorporar al codi base.

7.1.4 Afegir buscadors

Tant a la pantalla del mapa com a la de les meves "HealthSites", es va decidir incorporar un buscador a la respectiva "Action bar".

En el cas del mapa et serveix per buscar ràpidament una "HealthSite" i accedir al seu perfil. I en l'altre cas, pots buscar una "HealthSite" de la teva llista de seguides. Una nova funcionalitat a l'abast de l'usuari i que li facilita la feina a l'hora de trobar allò que busca. A més a més, visualment es va acomodar perquè no molestés i anés en concordança amb les pantalles.



(a) Map (b) Les meves HealthSites

Figure 56: Buscadors de HealthSites

7.1.5 No utilitzar notificacions Push

Les notificacions push són missatges que s'envien directament als dispositius mòbils, és a dir què és el servidor qui inicia la comunicació. Representen una manera moderna de fer arribar informació i tenen una gama àmplia d'opcions de personalització, a diferència dels texts plans, que poden augmentar la participació o interacció dels usuaris.

Però, tot i que a la primera versió de l'aplicació sí que es va implementar una notificació "push", aquesta no era rellevant perquè realment no li servia a l'usuari, i per tant no s'han implementat.

No se li veia el sentit a enviar notificacions, per exemple, quan iniciaves sessió, o quan seguies una "Healthsite", o quan comentaves, etc. No hi havia una acció clara que hagués d'estar recolzada per una notificació "push", o cap event que n'hagués de disparar alguna.

En un futur, quan les "HealthSites" puguin afegir entrades o actualitzacions de contingut, no

decau la idea de fer arribar notificacions "push" a tots els usuaris que la segueixin. En aquest cas sí que se li veu un sentit.

7.2 Desenvolupament de l'aplicació

Aquesta secció es centrarà a explicar certes parts importants que s'han desenvolupat seguint el disseny establert basat en el catàleg de requeriments, com i quines dificultats s'han trobat en el procés.

7.2.1 Firebase Authentication

És un servei de *back-end* proporcionat per *Firebase*, que permet autenticar als usuaris d'una aplicació. Té moltes opcions, i permet autenticar-se mitjançant contrasenyes, números de telèfon, proveïdors d'identitat federada, com *Google*, *Facebook* i *Twitter*, entre altres.

L'aplicació haurà de recuperar les credencials específiques de l'usuari que es vol autenticar, aleshores s'hauran de passar les credencials al SDK de *Firebase Authentication* i finalment es verificaran retornant una resposta la qual podrem gestionar internament.

Pel que fa a la implementació, es recuperen les dades de la pantalla d'inici de sessió en el moment que es prem el botó d'entrar. Aleshores mitjançant l'SDK de *Firebase Authentication*, concretament el mètode "signiInWithEmailAndPassword(email, password)" es verifiquen les dades d'accés. Per tant si aquestes són correctes se li dòna pas a l'usuari; si no ho són, es mostra un missatge d'error.

```
if(validLogInForm()) {
    showProgress()
    mAuth!!.signInWithEmailAndPassword(email.text.toString(), password.text.toString())
        .addOnSuccessListener { it: AuthResult!
            startMapActivity()
        }
        .addOnFailureListener { it: Exception
            val toast : Toast! = Toast.makeText(applicationContext, "Authentication failed", Toast.LENGTH_SHORT)
            toast.setGravity(Gravity.TOP,  xOffset: 0,  yOffset: 40)
            toast.show()
            hideProgress()
        }
}
```

Figure 57: Codi relatiu a l'autenticació de Firebase

7.2.2 Fragments

Què és un Fragment? És una part de la interfície d'usuari que forma part internament d'una Activitat.

Es poden combinar diversos fragments en una mateixa pantalla i activitat, a més a més de reutilitzar-lo per a altres.

Tenen un cicle de vida propi, però en haver d'estar sempre allotjats dins d'una activitat el seu cicle també és dependent del de l'activitat amfitriona. Per exemple, quan l'activitat està en pausa, els seus fragments també. Però mentrestant una activitat s'està executant, es poden manipular els fragments de forma independent a la pròpia conveniència.

Dins l'aplicació s'han utilitzat diversos fragments per implementar algunes funcionalitats i no estar constantment creant activitats i passant dades entre aquestes, fet que redueix l'eficiència i el rendiment en haver de gestionar internament tantes tasques. Seguidament trobareu diverses subseccions comentant las parts on s'han utilitzat fragments.

7.2.2.1 Pantalla principal

En la pantalla principal s'utilitzen diversos fragments per implementar certes parts de la seva interfície. A causa d'usar la "bottom navigation" s'havia d'apostar pels fragments entre les diferents pantalles necessàries per completar les opcions de navegació proporcionades. Les parts fragmentades són: el mapa, els filtres, la llista de "HealthSites" seguides i el perfil.

Totes aquestes parts són gestionades internament i quan se requereix la seva visibilitat doncs s'animen per aparèixer a disposició de l'usuari, tot just quan aquest prem la corresponent opció en la "bottom navigation".

Es va haver de refactoritzar i refer tota la part del codi corresponent al mapa per integrar-lo al nou fragment que es volia implementar. Anteriorment el mapa era una activitat sencera, però a causa del nou disseny gràfic es va haver de redissenyar el codi intern.

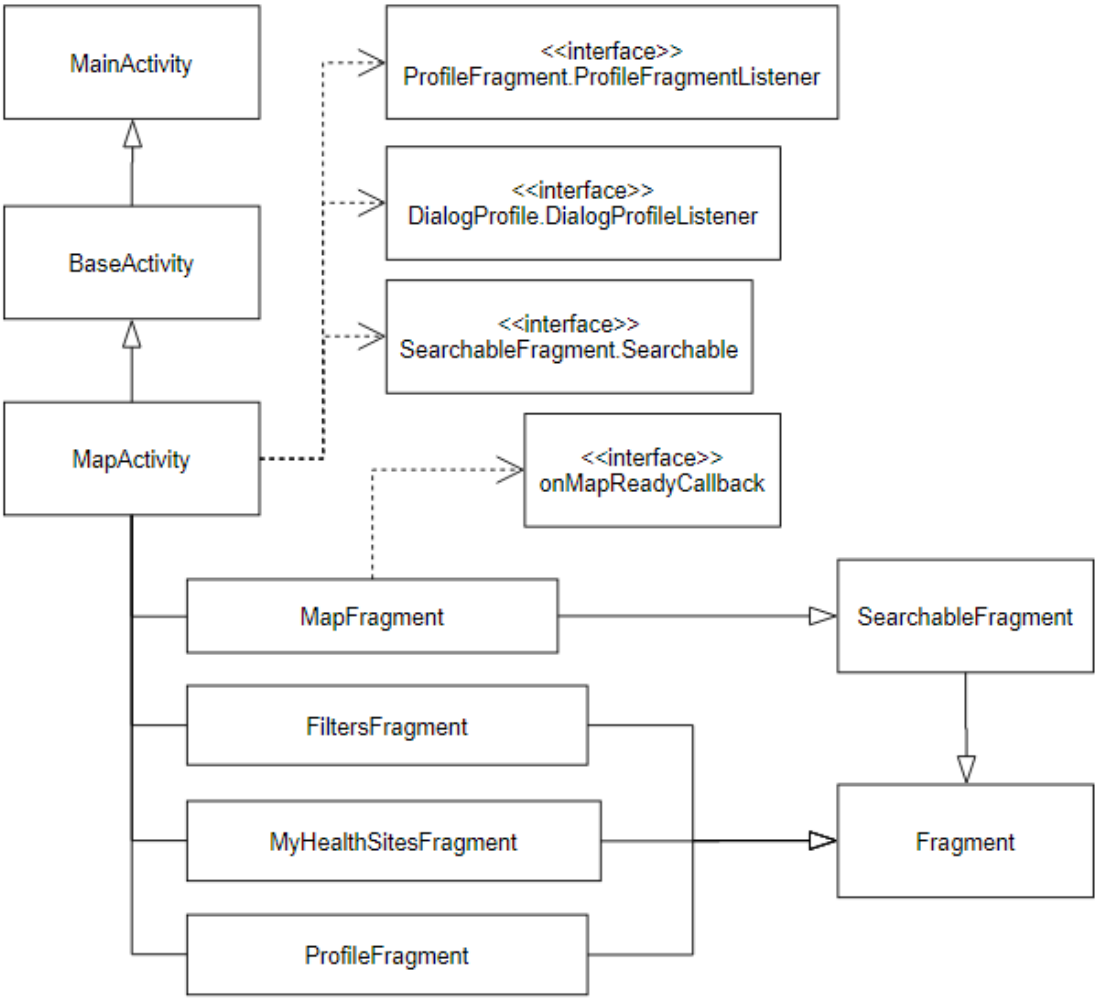


Figure 58: Diagrama UML de l'activitat principal

La "MapActivity" seria l'activitat principal que contindria els diferents fragments. La "BaseActivity" realitza accions de gestió de serveis com ara la disponibilitat de "GooglePlayServices" entre altres, i s'ha establert aquesta jerarquia per extreure aquestes funcionalitats de l'activitat principal.

Cadascun dels fragments hereten de la classe base Fragment tota la seva funcionalitat, menys el "MapFragment" en el qual s'implementa un buscador i és per això que ho fa a través del "SearchableFragment".

A més a més, el "MapFragment" implementa la funcionalitat del mapa a través de la interfície "onMapReadyCallback".

Finalment, la mateixa "MapActivity" implementa certes interfícies. La primera s'utilitza per comunicar el "ProfileFragment" amb l'activitat. La segona s'usa per, a través dels respectius "Dialogs", actualitzar les dades de l'usuari i comunicar-se amb l'activitat per realitzar l'actualització de les dades al *back-end*. I per acabar, l'última s'utilitza per inicialitzar el buscador i recuperar dades de certes estructures que guarden les "HealthSites" afegides al mapa per poder-les mostrar segons la cerca feta.

7.2.2.2 Formularis

Com s'ha comentat en seccions anteriors no es volien mantenir els típics formularis, estàtics, i d'una sola pantalla. Es volia donar dinamisme. I l'única opció era utilitzar "fragments" per construir les diferents seccions, ja que anar passant dades entre activitats que simplement haurien d'implementar una petita part del mateix formulari és poc eficient.

D'aquesta manera una sola activitat gestiona els fragments, i pel que fa a les dades s'estableix un canal de comunicació mitjançant interfícies per passar-les dels respectius fragments a l'activitat on es guarden fins al final. A més a més cal dir que les interfícies són exemples d'accions en favor de la reutilització del codi donat que així es redueix l'acoblament entre classes, un aspecte molt important en la millora del disseny i la codificació del codi per tal de fer-lo més fàcilment modificable (aquests és un dels "RNF" establerts inicialment).

En les següents seccions es realitza una petita explicació del disseny de cadascun dels formularis que s'ha implementat.

7.2.2.2.1 Registre d'usuari En la següent figura es pot veure la disposició de les classes i de la interfície usada per a la comunicació. També es pot veure la jerarquia que algunes mantenen entre si.

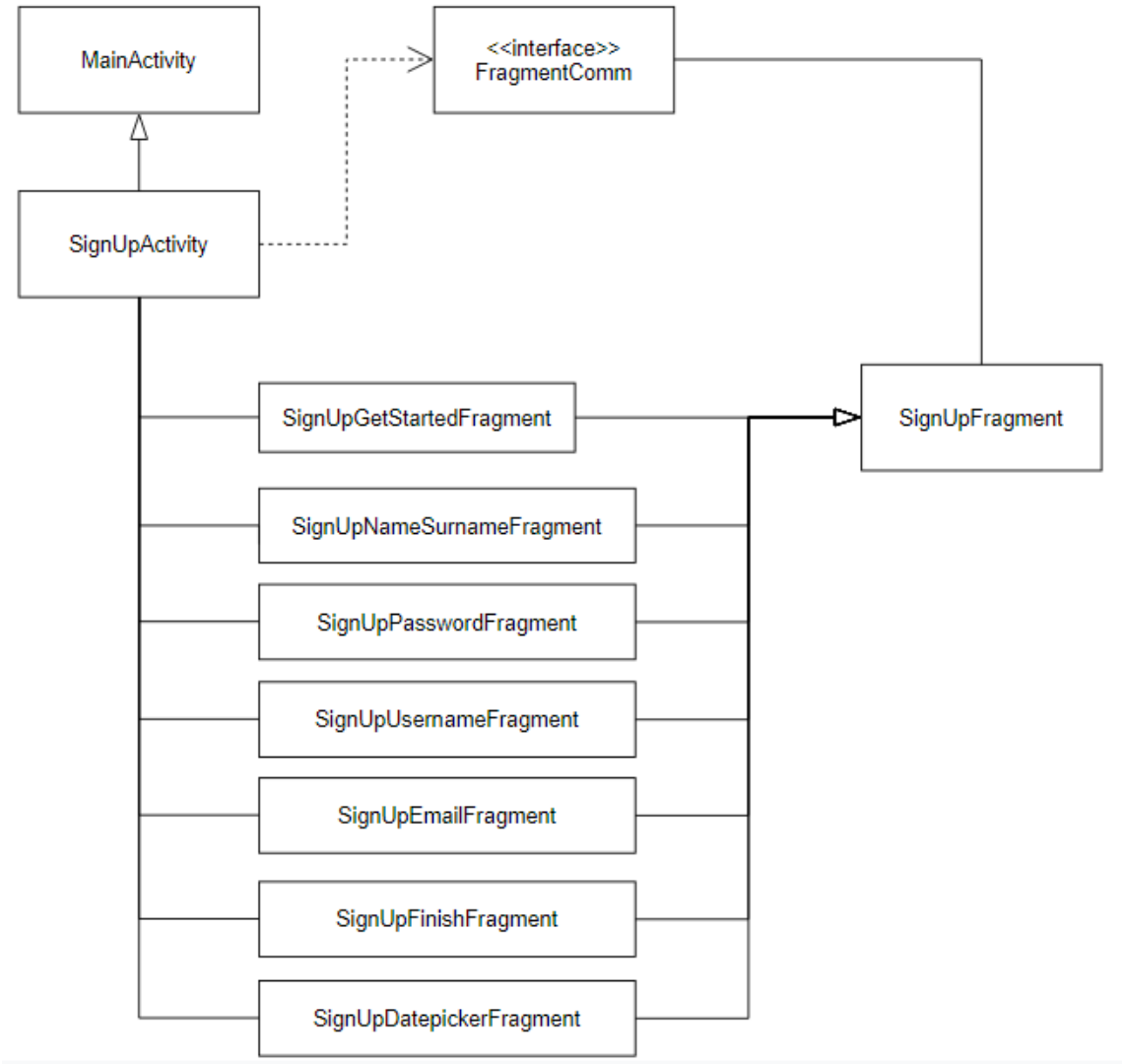


Figure 59: Diagrama UML del registre d'usuari

La classe abstracta "SignUpFragment" defineix els mètodes base que necessiten els fragments, a més a més de definir la interfície per a la comunicació. L'activitat implementa la interfície i cadascun dels fragments a través del context on s'inflen, és a dir, l'activitat crida als mètodes per realitzar la comunicació.

```
override fun onAttach(context: Context) {
    super.onAttach(context)

    listener = if (context is addFragmentListener) {
        context
    } else {
        throw RuntimeException("$context interface addFragmentListener not implemented")
    }
}
```

Figure 60: Mètode "onAttach()" dels fragments

En el mètode "onAttach()" (mostrat a la figura 60), el qual s'activa quan el fragment és carregat per primera vegada, es captura el context, és a dir l'activitat, i s'estableix el *listener* per quan faci falta procedir a la comunicació de dades.

Per acabar, a través de l'API de *Firebase* es fa el registre de l'usuari utilitzant el mètode "createUserWithEmailAndPassword" i si tot es fa correctament es guarden a la BBDD les dades de l'objecte "User" que s'ha anat construint durant tot el procés del formulari a través de la comunicació entre els fragments i l'activitat.

```
override fun signUp() {
    showProgress()
    FirebaseAuth.getInstance().createUserWithEmailAndPassword(user!!.email, user!!.password)
        .addOnSuccessListener { it: AuthResult!
            user?.setId(FirebaseAuth.getInstance().currentUser!!.uid)
            val map: Map<String, Any> = jacksonObjectMapper().convertValue(user!!)
            FirebaseFirestore.getInstance()
                .collection( collectionPath: "Users")
                .document(FirebaseAuth.getInstance().currentUser!!.uid).set(map)
                .addOnSuccessListener { it: Void!
                    Log.i( tag: "Set User",  msg: "User Registration OK")
                    Toast.makeText(
                        applicationContext,
                        getString(R.string.user_register_ok),
                        Toast.LENGTH_SHORT).show()
                    startMapActivity()
                }.addOnFailureListener { it: Exception
                    Log.e( tag: "Set User",  msg: "User Registration KO")
                    Toast.makeText(
                        applicationContext,
                        "User Registration KO",
                        Toast.LENGTH_SHORT).show()
                }
        }.addOnFailureListener { exception: Exception ->
            Log.e( tag: "Fail signup",  msg: "createUserWithEmail:failure", exception)
            hideProgress()
            Toast.makeText(
                applicationContext,
                getString(R.string.user_register_ko),
                Toast.LENGTH_SHORT).show()
        }
}
```

Figure 61: Mètode per crear un nou usuari

7.2.2.2.2 Registre d'una nova HealthSite Aquest cas és igual a l'explicat en l'anterior secció. Únicament volia deixar especificat el seu diagrama de forma gràfica. I també el mètode utilitzat per crear una nova "HealthSite" a partir del respectiu objecte construït al llarg de tot el formulari.

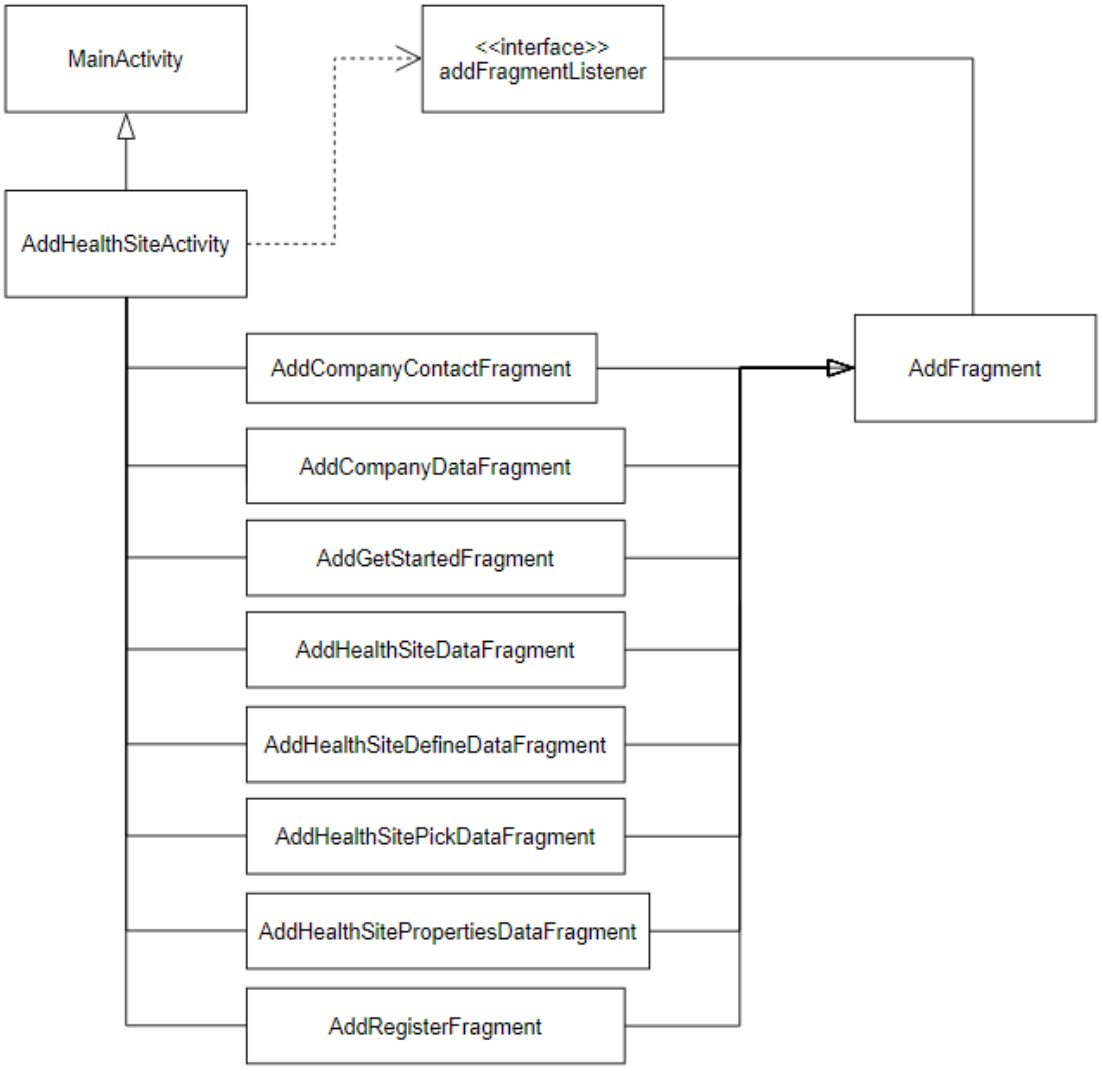


Figure 62: Diagrama UML del registre d'una HealthSite

```
override fun register() {
    showProgress()
    FirebaseFirestore.getInstance()
        .collection( collectionPath: "HealthSites")
        .whereEqualTo( field: "companyCIF", healthSite.companyCIF).get()
        .addOnCompleteListener { task : Task<QuerySnapshot!> ->
            if(task.isSuccessful && task.result!!.documents.size == 0) {
                val id : String = FirebaseFirestore.getInstance()
                    .collection( collectionPath: "HealthSites")
                    .document().id
                this.healthSite.setId(id)
                val map: Map<String, Any> = jacksonObjectMapper().convertValue(this.healthSite)
                FirebaseFirestore.getInstance()
                    .collection( collectionPath: "HealthSites")
                    .document(id).set(map)
                    .addOnCompleteListener { task : Task<Void!> ->
                        if (task.isSuccessful) {
                            handleUpload(image!!, id)
                            Toast.makeText(
                                applicationContext,
                                getString(R.string.healthsite_register_ok),
                                Toast.LENGTH_SHORT).show()
                        } else if (task.isCanceled) {
                            hideProgress()
                            Toast.makeText(
                                applicationContext,
                                getString(R.string.healthsite_register_ko),
                                Toast.LENGTH_SHORT).show()
                        }
                    }
            } else {
                hideProgress()
                Toast.makeText( context: this,
                    text: "This company CIF is already taken",
                    Toast.LENGTH_SHORT).show()
            }
        }
}
```

Figure 63: Mètode per crear una nova HealthSite

7.2.3 MultiWatchers

Un "watcher" és simplement aquell codi que estableix una *callback* per capturar un cert event en un element de la interfície. Per tant un "MultiWatcher" ho farà amb tots els seus elements registrats.

S'han implementat dos tipus de "MultiWatchers", un per a "EditTexts" i un altre per a les "CheckBox", perquè els requeriments d'algunes parts a implementar així ho exigien.

Per exemple, en els formularis s'havia d'utilitzar el corresponent als "EditText" per saber quan l'usuari podria avançar a la següent part. I per altra banda, l'altre s'usa en el registre d'una nova "HealthSite" quan s'han d'especificar les seves propietats.

```
class MultiTextWatcher {
    private var callback: TextWatcherWithInstance? = null

    fun setCallback(callback: TextWatcherWithInstance?) {
        this.callback = callback
    }

    fun registerEditText(editText: EditText): MultiTextWatcher {
        editText.addTextChangedListener(object : TextWatcher {
            override fun beforeTextChanged(s: CharSequence, start: Int, count: Int, after: Int) {
                callback!!.beforeTextChanged(editText, s, start, count, after)
            }

            override fun onTextChanged(s: CharSequence, start: Int, before: Int, count: Int) {
                callback!!.onTextChanged(editText, s, start, before, count)
            }

            override fun afterTextChanged(editable: Editable) {
                callback!!.afterTextChanged(editText, editable)
            }
        })
        return this
    }

    interface TextWatcherWithInstance {
        fun beforeTextChanged(editText: EditText?, s: CharSequence?, start: Int, count: Int, after: Int)
        fun onTextChanged(editText: EditText?, s: CharSequence?, start: Int, before: Int, count: Int)
        fun afterTextChanged(editText: EditText?, editable: Editable?)
    }
}
```

Figure 64: Classe MultiTextWatcher

```
MultiTextWatcher()
    .registerEditText(signup_name)
    .registerEditText(signup_surname)
    .setCallback(object : MultiTextWatcher.TextWatcherWithInstance {
        override fun beforeTextChanged(editText: EditText?, s: CharSequence?, start: Int, count: Int, after: Int) { }

        override fun onTextChanged(editText: EditText?, s: CharSequence?, start: Int, before: Int, count: Int) { }

        override fun afterTextChanged(editText: EditText?, editable: Editable?) {
            if (!TextUtils.isEmpty(signup_name.text.toString()) && !TextUtils.isEmpty(signup_surname.text.toString())) {
                btn_next_name_surname.alpha = 1f
                btn_next_name_surname.isEnabled = true
            } else {
                btn_next_name_surname.alpha = 0.3.toFloat()
                btn_next_name_surname.isEnabled = false
            }
        }
    })
```

Figure 65: Exemple d'ús del MultiTextWatcher

7.2.4 Custom PlacePicker

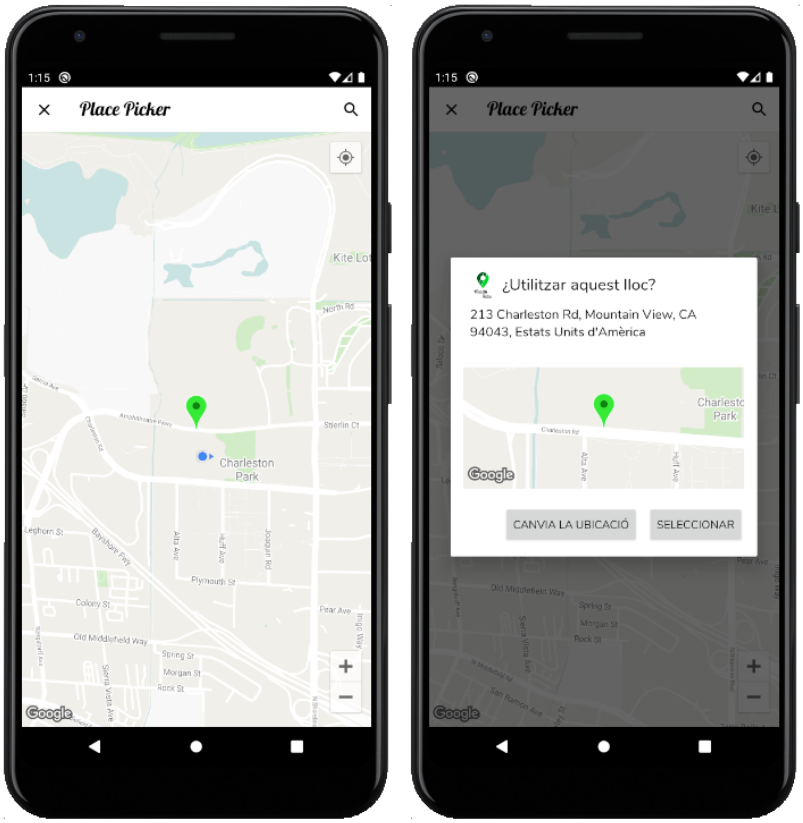
La classe "PlacePicker" que s'havia utilitzat fins ara està *deprecated*. Significa que ja no és la manera correcta de procedir i que deixa de tenir suport perquè pot contenir errors i falles de seguretat.

public class **PlacePicker** extends Object

This class is deprecated.
The Google Play Services Places SDK is deprecated. A new SDK is available. See the client migration guide for more information.

Figure 66: Class PlacePicker deprecated

Aleshores es va desenvolupar un "PlacePicker" personalitzat tot de nou, i únicament per a aquesta aplicació.



(a) Pantalla inicial (b) Selecció realitzada

Figure 67: Pantalla del PlacePicker personalitzat

És una activitat que implementa "OnMapReadyCallback" per poder mostrar un mapa en el qual es fa la selecció corresponent de l'adreça.

A més a més s'hi va incorporar un buscador de llocs utilitzant *Google Places*, tot i que actualment no estarà disponible perquè s'ha d'activar la facturació en el compte de *Google*.

En resum, s'extreu l'adreça completa i la geolocalització del lloc que selecciona l'usuari. A continuació es podrà observar com s'ha establert el *listener* per actuar sobre el click en el mapa i recuperar l'adreça.

```
mMap.setOnMapClickListener((latLng) → {
        MarkerOptions markerOptions = new MarkerOptions();
        markerOptions.position(latLng);
        markerOptions.title(getAddress(latLng));
        markerOptions.icon(BitmapDescriptorFactory.defaultMarker(BitmapDescriptorFactory.HUE_GREEN));

        CameraUpdate location = CameraUpdateFactory.newLatLngZoom(latLng,  v: 15);
        mMap.clear();
        mMap.animateCamera(location);
        mMap.addMarker(markerOptions);
});
```

Figure 68: Listener per recuperar l'adreça

```
private String getAddress(LatLng latLng) {

    Geocoder geocoder;
    List<Address> addresses;
    geocoder = new Geocoder( context: this, Locale.getDefault());

    try {
        addresses = geocoder.getFromLocation(latLng.latitude, latLng.longitude,  maxResults: 1);

        String address = addresses.get(0).getAddressLine( index: 0);
        String country = addresses.get(0).getCountryName();
        String countryCode = addresses.get(0).getCountryCode();
        String city = addresses.get(0).getLocality();
        String adminArea = addresses.get(0).getAdminArea();
        String postalCode = addresses.get(0).getPostalCode();
        String street = addresses.get(0).getThoroughfare();
        String number = addresses.get(0).getSubThoroughfare();

        new DialogSelectPlacePicker(latLng.latitude,
                latLng.longitude, address, countryCode, country, adminArea,
                city, postalCode, street, number)
                .show(getSupportFragmentManager(),  tag: "dialogSelectPlacePicker");

        return address;
    } catch (IOException e) {
        e.printStackTrace();
        Toast.makeText( context: this, "Check your connection", Toast.LENGTH_SHORT).show();
        return "No Address Found";
    }
}
```

Figure 69: Funció per extreure l'adreça del lloc seleccionat

7.2.5 Marker Clustering

S'ha incorporat una nova funcionalitat anomenada clúster de marcadors. Ens ajuda a gestionar els múltiples marcadors a través de diferents nivells de zoom del mapa. Realment, els marcadors es tracten com a ítems i solament esdevenen marcadors quan el nivell de zoom és l'adequat per renderitzar-los. D'aquesta manera aconseguirem agrupar els marcadors pròxims i tenir un mapa més net, a part que donarem informació valuosa a l'usuari sobre quants marcadors hi ha en una determinada àrea.

Els ítems en el mapa són les "HealthSites", per tant la seva classe implementarà la interfície "ClusterItem". Llavors haurem d'afegir cadascuna de les "HealthSites" del mapa a l'objecte "ClusterManager" encarregat de la gestió de tot plegat. El "ClusterManager" passa els marcadors al seu algoritme intern que els transforma en un grup de clústers.

A més a més, si es vol utilitzar un renderitz-ador personalitzat aquest també se li pot passar al "ClusterManager" com és el cas.

Addicionalment, s'ha afegit també al "ClusterManager" un personalitzador de la finestra d'informació que es visualitza quan es pitja en un marcador.

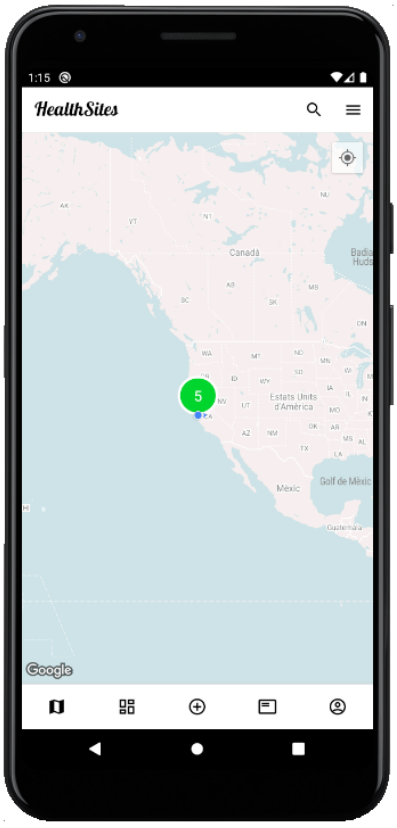


Figure 70: Icona del clúster de marcadors

Figure 71: Finestra d'informació personalitzada

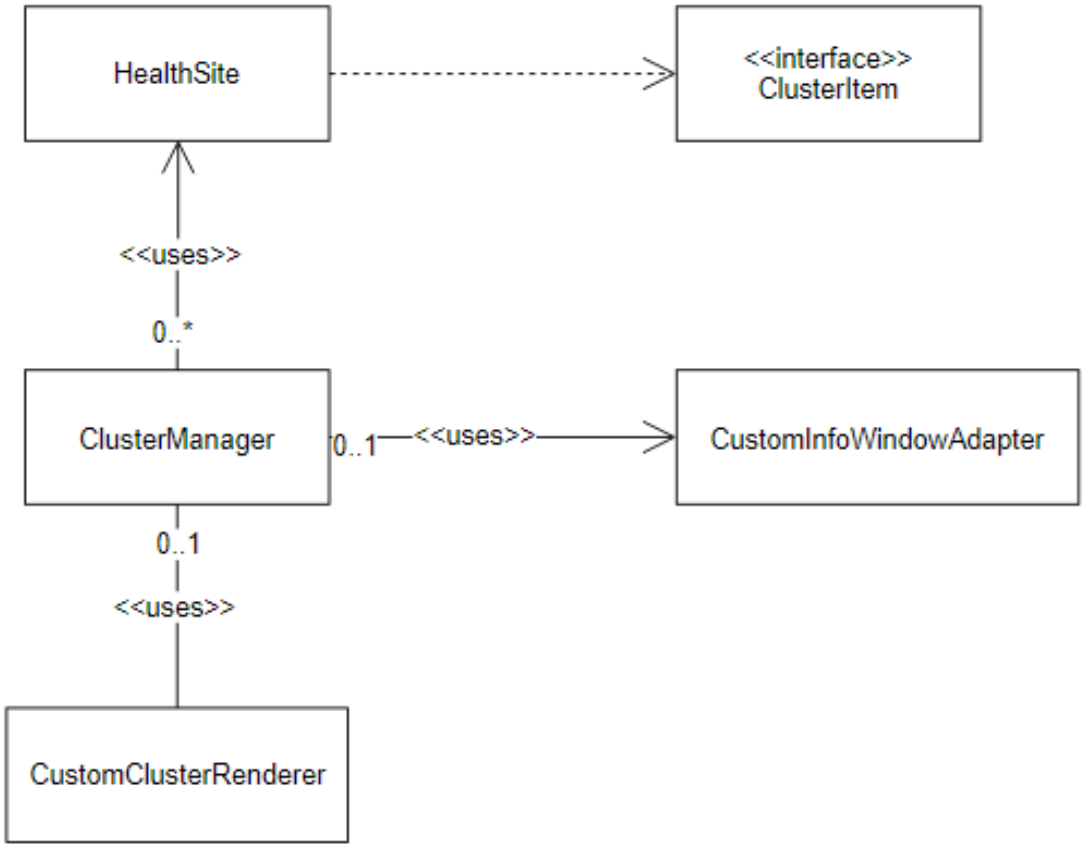


Figure 72: Diagrama UML del clúster de marcadors

7.2.6 Animacions

Les animacions entre pantalles o fragments, o aplicades a elements específics s'han definit utilitzant fitxers .xml en el subdirectori /res/anim del projecte, seguint les recomanacions de la guia per desenvolupadors.

```xml
<set xmlns:android="http://schemas.android.com/apk/res/android"
    android:duration="@android:integer/config_mediumAnimTime"
    android:interpolator="@android:anim/decelerate_interpolator">
    <alpha
        android:fromAlpha="0"
        android:toAlpha="1" />
</set>
```

(a) Animació *fade-in*

```xml
<set xmlns:android="http://schemas.android.com/apk/res/android"
    android:duration="@android:integer/config_mediumAnimTime"
    android:interpolator="@android:anim/decelerate_interpolator">
    <translate
        android:fromXDelta="100%"
        android:toXDelta="0%" />
</set>
```

(b) Animació *slide-in-from-right*

Figure 73: Mostra de fitxers XML de les animacions

Per exemple, en les següents imatges podreu observar com s'utilitzen les animacions en el codi per implementar les animacions desitjades. En el primer cas es realitza una transició animada entre dues activitats especificant les opcions; i en el segon entre fragments afegint les requerides a la transacció.

```
startActivity(
    Intent( packageContext: this,
        SettingsActivity::class.java),
    ActivityOptions.makeCustomAnimation(
        applicationContext,
        R.anim.slide_in_from_right,
        R.anim.slide_out_to_left)
    !!.toBundle())
```

(a) Animació entre activitats

```
supportFragmentManager
    .beginTransaction()
    .setCustomAnimations(
        R.anim.fade_in,
        R.anim.fade_out,
        R.anim.slide_in_from_right,
        R.anim.fade_out)
    .replace(R.id.container_fr,
        SignUpGetStartedFragment(),
        tag: "get_started")
    .addToBackStack( name: null)
    .commit()
```

(b) Animació entre fragments

Figure 74: Mostra de codi aplicant animacions de transició

7.2.7 Multi-Idioma

A la pantalla dels ajustaments tens la possibilitat de canviar el idioma elegint entre 4 possibilitats: Anglès (idioma per defecte), Castellà, Català, o idioma del dispositiu.

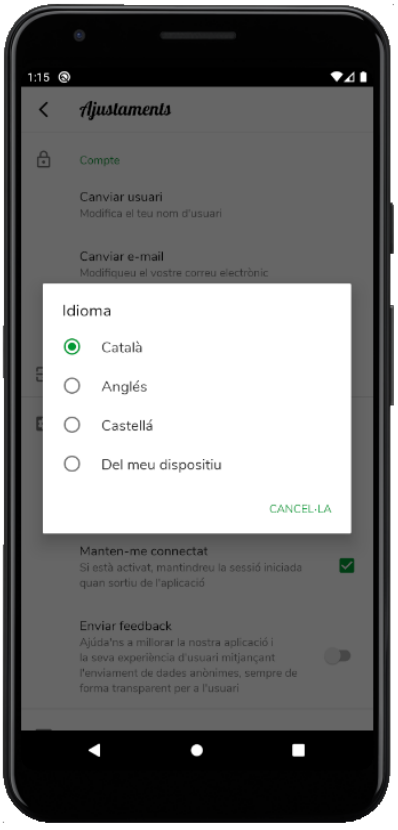


Figure 75: Selecció de l'idioma als ajustaments

Si tries qualsevol dels tres primers, se't mostrarà l'idioma seleccionat. No obstant, si tries l'última opció, idioma del dispositiu, aleshores si tens el mòbil en algun dels tres primers idiomes doncs se't posarà aquell directament, però si tens el mòbil en un idioma no disponible a l'aplicació se t'establirà l'Anglès al ser l'idioma per defecte.

```
open class MainActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setLocale()
    }

    @Suppress( ...names: "DEPRECATION")
    protected fun setLocale() {
        val lang : String?  = PreferenceManager
            .getDefaultSharedPreferences( context: this)
            .getString( key: "pref_language",  defValue: "")
        if (!lang!!.isEmpty()) {
            val locale = Locale(lang)
            val config = Configuration()
            val dm : DisplayMetrics!  = baseContext.resources.displayMetrics
            Locale.setDefault(locale)
            config.locale = locale
            baseContext.resources.updateConfiguration(config, dm)
        }
    }
}
```

Figure 76: Codificació de l'idioma

Per implementar aquesta funcionalitat es va jerarquitzar les activitats i totes elles hereten de la classe abstracta "MainActivity". D'aquesta manera, en els respectius "onCreate()" s'assegura què s'aplicarà la funció "setLocale()" gràcies a les crides "super()". I així quan s'inicia una activitat aquesta ja aplicarà l'idioma corresponent a la configuració.

Per altra banda, quan a la pantalla d'ajustaments es selecciona una opció es fa una crida a la funció "setLocale()" directament perquè s'hereta al tenir el modificador "protected".

Finalment, a l'activitat "SettingsActivity" es modifica el llenguatge a la configuració quan es canvia i es recrea l'activitat per a carregar el nou.

```
fun setLocale(lang: String) {
    val locale = Locale(lang)
    val config = Configuration()
    val dm: DisplayMetrics = baseContext.resources.displayMetrics

    Locale.setDefault(locale)
    config.locale = locale
    baseContext.resources.updateConfiguration(config, dm)

    recreate()
}
```

Figure 77: Funció per canviar l'idioma i recrear l'activitat actual

7.2.8 Filtres

La pantalla de filtres és un fragment, per això la seva funcionalitat es gestiona internament en el codi del fragment.

Per implementar els filtres, es sabia que la disposició s'havia de guardar per mantenir-la. En resum, s'ha de guardar a les preferències del sistema per tal que quan aquest torni a la pantalla de filtres s'ho trobi igual com ho havia deixat.

Llavors, es manté un recull de camps cadascun corresponent a un filtre especific a les corresponents "SharedPreferences". L'API de "SharedPreferences" inclosa en el SDK d'*Android*, conté un objecte "SharedPreferences" que apunta directament a un fitxer que conté parells de clau-valor i proporciona mètodes per llegir, emmagatzemar i actualitzar els valors. D'aquesta forma no només es poden aplicar els canvis en el fragment del mapa per mostrar els marcadors filtrats, sinó que al reobrir els filtres es pot carregar la seva disposició.

Quan una targeta és pitjada s'actualitza el valor del filtre i s'estableix a positiu un camp de control que també es guarda a les preferències per saber que s'han produït canvis. I al retornar a la pantalla del mapa, es comprova si hi ha hagut algun canvi gràcies al camp de control, i en cas afirmatiu doncs es buida el mapa i es procedeix a carregar els nous marcadors segons les propietats filtrades.

```
vegan!!.setOnClickListener { v : View! ->
    val editor : SharedPreferences.Editor! = mPrefs!!.edit()
    if (!mPrefs!!.getBoolean( key: "vegan_filter", defValue: false)) {
        editor.putBoolean("vegan_filter", true)
        vegan_filter_layout.setBackgroundResource(R.drawable.bg_autocomplete)
        v.elevation = 1f
        veganCheck2 = true
    } else {
        editor.putBoolean("vegan_filter", false)
        vegan_filter_layout.setBackgroundResource(0)
        v.elevation = 22f
        veganCheck2 = false
    }
    editor.apply()
    setPrefFiltersChanged()
}
```

Figure 78: Codi relatiu al *listener* d'un filtre concret

Quan es dispara el *listener* en el moment que l'usuari fa clic, doncs s'edita la preferència "vegan_filter" en aquest cas. S'actualitza la interfície de la manera corresponent, s'apliquen els canvis, i s'actualitza el control dels canvis dels filtres.

7.2.9 ViewPager

És un element que ens permet desplaçar-nos horitzontalment a través de diferents fragments dins d'una mateixa activitat.

S'ha utilitzat per aplicar-ho al perfil de la "HealthSite" (mostrat a la figura 37) on havíem de mostrar certes dades (Propietats, Contacte, Comentaris) però no podia estar tot amuntegat. Aleshores es va decidir incorporar aquest control que juntament amb unes "tabs" són els encarregats de gestionar quina informació es mostra segons la interacció de l'usuari.

```
viewPager = findViewById(R.id.view_pager);
tabLayout = findViewById(R.id.tab_layout);

tabLayout.setupWithViewPager(viewPager);

ViewPagerAdapter viewPagerAdapter = new ViewPagerAdapter(getSupportFragmentManager(),  behavior: 0);
viewPagerAdapter.addFragment(infoFragment, "Properties");
viewPagerAdapter.addFragment(contactFragment, "Contact");
viewPagerAdapter.addFragment(postsFragment, "Posts");
viewPager.setAdapter(viewPagerAdapter);
```

Figure 79: Inicialització del ViewPager

```
private static class ViewPagerAdapter extends FragmentPagerAdapter {

    private List<Fragment> fragments = new ArrayList<>();
    private List<String> fragmentsTitles = new ArrayList<>();

    public ViewPagerAdapter(@NonNull FragmentManager fm, int behavior) { super(fm, behavior); }

    public void addFragment(Fragment fragment, String title) {
        fragments.add(fragment);
        fragmentsTitles.add(title);
    }

    @NonNull
    @Override
    public Fragment getItem(int position) {
        return fragments.get(position);
    }

    @Override
    public int getCount() { return fragments.size(); }

    @Nullable
    @Override
    public CharSequence getPageTitle(int position) { return fragmentsTitles.get(position); }
}
```

Figure 80: Classe del ViewPagerAdapter

7.2.10 Sincronització de dades

En parts del codi s'han implementat els anomenats "SnapshotListeners", els quals serveixen per recuperar dades quan aquestes han patit algun canvi.

Per exemple, al perfil de la "HealthSite" pots seguir-la o deixar-la de seguir, veritat? Doncs quan interactua l'usuari ràpidament el comptador de seguidors d'aquesta s'actualitza perquè s'ha establert un *listener* en una referència a la BBDD específica.

```
reference.addSnapshotListener((value, error) → {
        if(value != null) {
            if(value.contains(userId)) {
                followCounts = Objects.requireNonNull(value.getData()).size();
                btnFollow.setBackground(getDrawable(R.drawable.btn_outline_rounded));
                btnFollow.setText("Following");
                btnFollow.setTextColor(getResources().getColor(R.color.colorGreen));
                followCounter.setText(Integer.toString(followCounts));
                followChecker = true;
            } else {
                if(value.getData()!= null) {
                    followCounts = value.getData().size();
                } else {
                    followCounts = 0;
                }
                btnFollow.setBackground(getDrawable(R.drawable.btn_rounded));
                btnFollow.setText("Follow");
                btnFollow.setTextColor(getResources().getColor(android.R.color.white));
                followCounter.setText(Integer.toString(followCounts));
                followChecker = false;
            }
        }
});
```

Figure 81: SnapshotListener per actualitzar els seguidors d'una HealthSite

8 Pla de negoci

L'estratègia es centraria inicialment en promocionar l'aplicació per a què els usuaris s'animin a afegir les seves botigues o restaurants. És a dir aconseguir construir una bona BBDD. Hauria de ser el punt clau en la campanya inicial.

Per aconseguir aquest fi, actualment hi ha moltes possibilitats. Es podria crear comptes oficials a les principals xarxes socials: *Instagram*, *Facebook*, etc. i a més a més afegir anuncis específics per a cadascuna d'elles. Específics, ja que els usuaris de la primera xarxa social acostumen a tenir un perfil més jove que en la segona, la qual es pot considerar d'un perfil una mica més professional. D'aquesta manera, es podria dissenyar campanyes dedicades per mostrar *Instagram* i *Facebook Adds* (Anuncis).

Altrament, es podria fer un seguiment de l'aplicació mitjançant Google Analytics què la mateix plataforma de *Firebase* ens permet incorporar-lo a la nostra aplicació. D'aquesta manera podríem realitzar estudis sobre les diferents dades recaptades.

Per altra banda, quan es comenci a fidelitzar usuaris es podria afegir anuncis no massa intrusius en la versió "free" i afegir una versió de pagament sense aquestos però amb un preu assequible, l'anomenada "premium".

Finalment, gràcies a l'anàlisi de la competència va sorgir la idea de realitzar una pàgina web per introduir als futurs usuaris i interessats tal com les aplicacions de la competència tenien, i què en un futur podria incorporar funcionalitats de suport. Seguidament s'hi fa menció.

8.1 Pàgina Web

S'ha realitzat una pàgina web multi-idioma, els mateixos que estan disponibles a l'aplicació: Anglès, Castellà i Català, per promocionar i donar a conèixer l'aplicació, així com per donar informació de la mateixa i establir una altra ruta de contacte i de suport.

Quan es va plantejar el pla de negoci, i havent vist que la competència tal com s'esmenta en la secció corresponent, disposava de pàgina web per a la seva aplicació, doncs es va veure amb molt bons ulls la idea de construir una pàgina web estàtica de suport al projecte.

Conjuntament amb la dissenyadora es va realitzar l'estructuració dels continguts i l'estil, seguint la paleta de colors de l'aplicació.

8.1.1 Tecnologia utilitzada

És una web realitzada amb "node.js" juntament amb el mòdul "express" que s'encarrega de disposar el servidor per fer córrer la web.

S'han utilitzat diversos mòduls de més, com ara "ejs", el qual serveix per renderitzar plantilles que incorporen codi .js en el seu interior. D'aquesta manera podem crear pàgines dinàmiques de contingut i reutilitzar parts en altres.

També s'utilitza "nodemailer" per crear un formulari de contacte i enllaçar-lo amb el servidor de correu.

Finalment pel que fa a les dependències s'han utilitzat "body-parser" i "nodemon". La primera s'utilitza conjuntament amb el mòdul "express" ja que serveix de *middleware* entre les sol·licituds i el servidor. I la segona, és una eina que ens ajuda a organitzar el projecte de node durant el desenvolupament.

En la següent fotografia es pot veure el fitxer de "package.json" respectiu a la configuració:

package.json

```
{
  "name": "healthsites",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "start": "node src",
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": ["heath", "healthy","vegan","vegetarian","ecologic","gluten", "latose", "map", "app"],
  "author": "albert pérez datsira",
  "license": "ISC",
  "dependencies": {
    "ejs": "^3.1.3",
    "express": "^4.17.1",
    "morgan": "^1.10.0",
    "nodemailer": "^6.4.11"
  },
  "devDependencies": {
    "body-parser": "^1.19.0",
    "nodemon": "^2.0.4"
  }
}
```

Figure 82: Fitxer de configuració de la web

8.1.2 Arbre del projecte

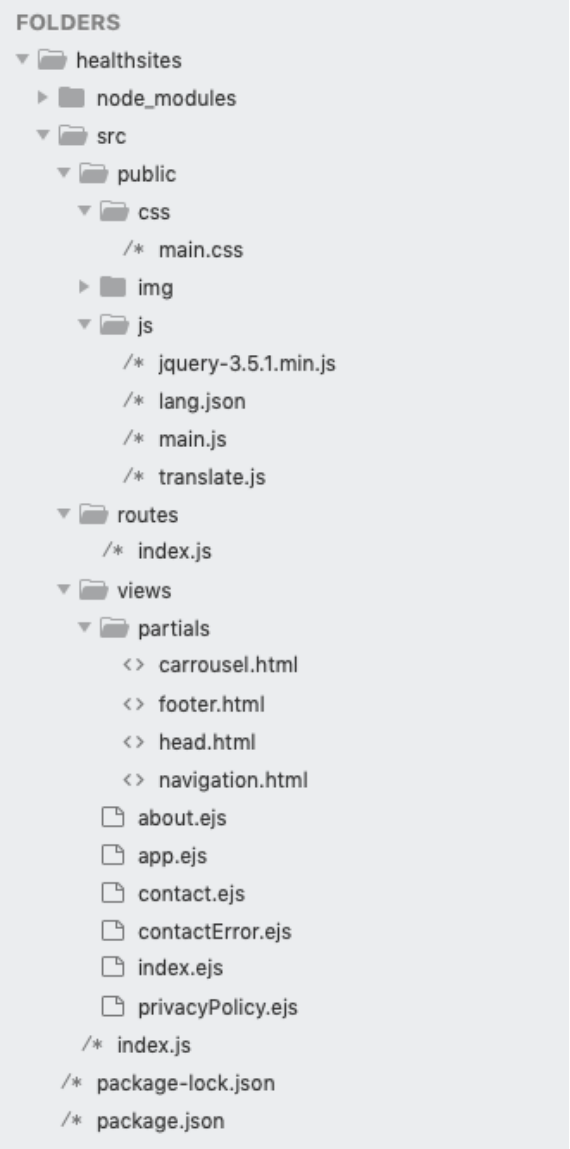


Figure 83: Arbre dels directoris i fitxers de la web

8.1.3 Pàgines

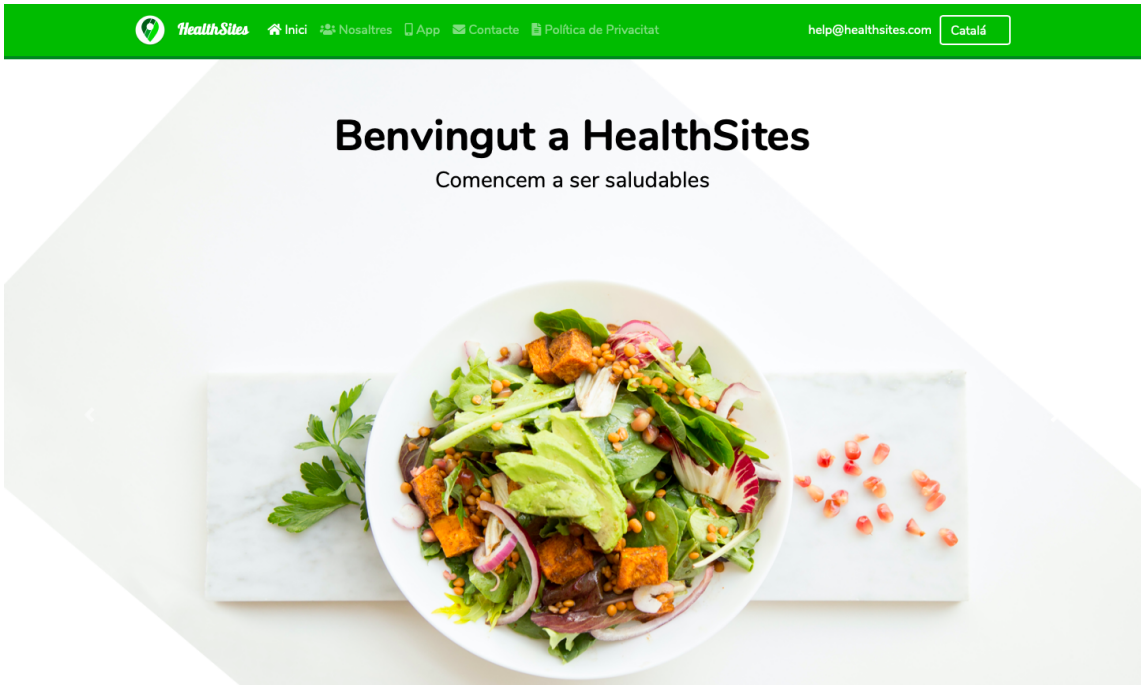


Figure 84: Diapositiva 1 de l'*slider* de la pàgina d'inici de la web

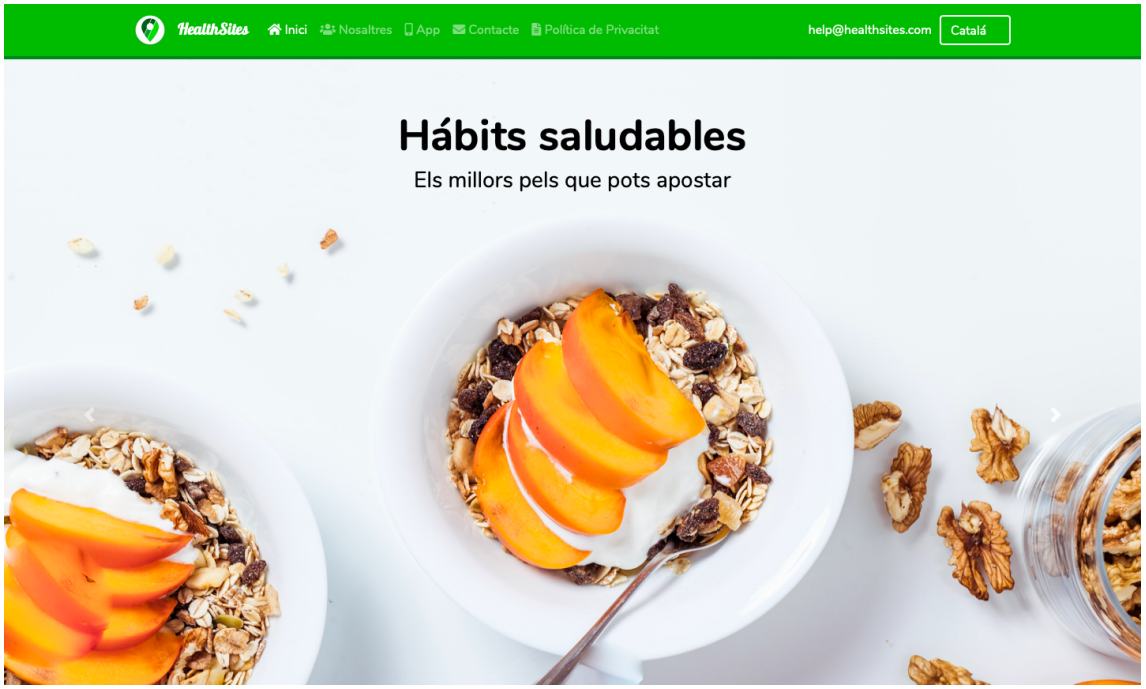


Figure 85: Diapositiva 2 de l'*slider* de la pàgina d'inici de la web

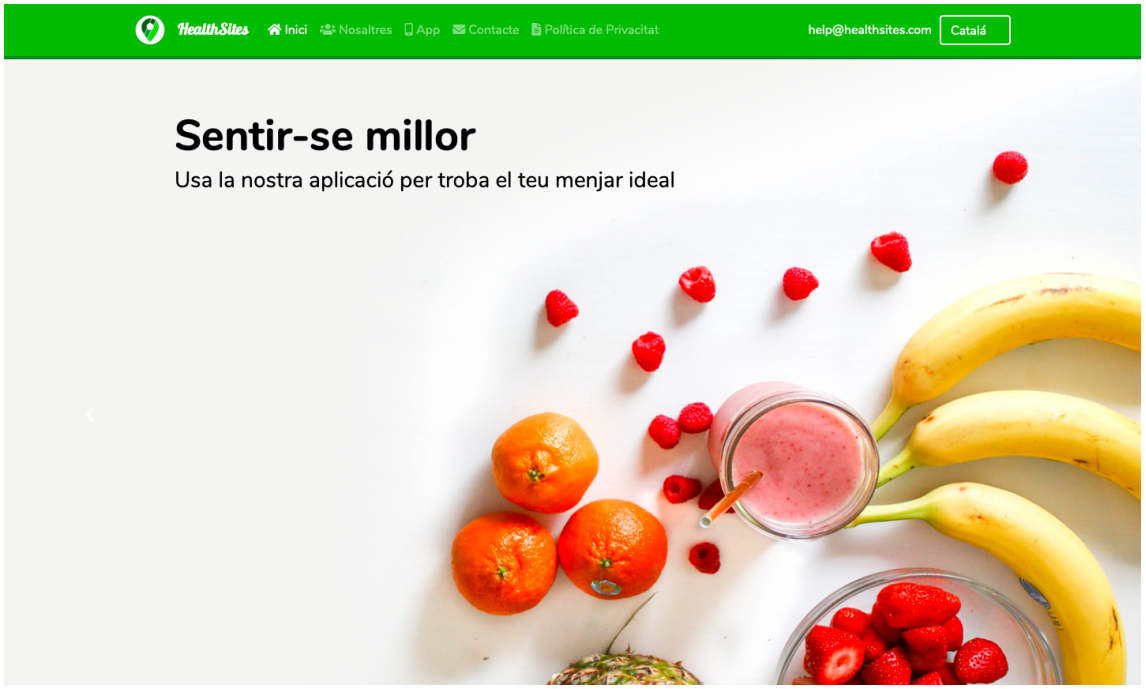


Figure 86: Diapositiva 3 de l'*slider* de la pàgina d'inici de la web

Figure 87: Inici part 1

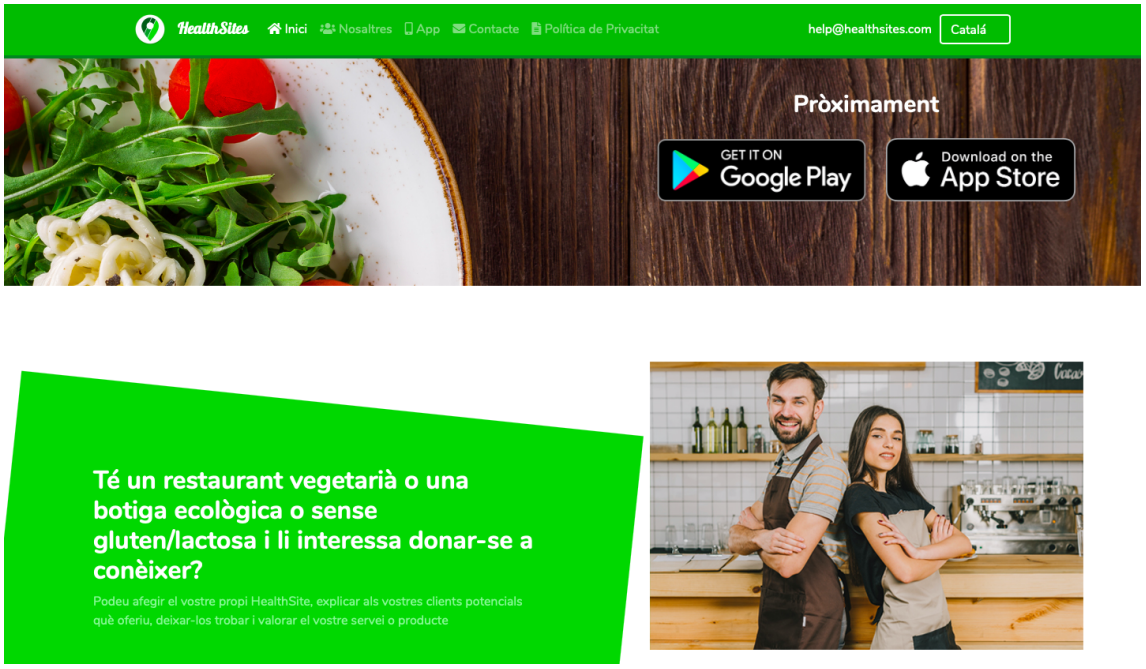
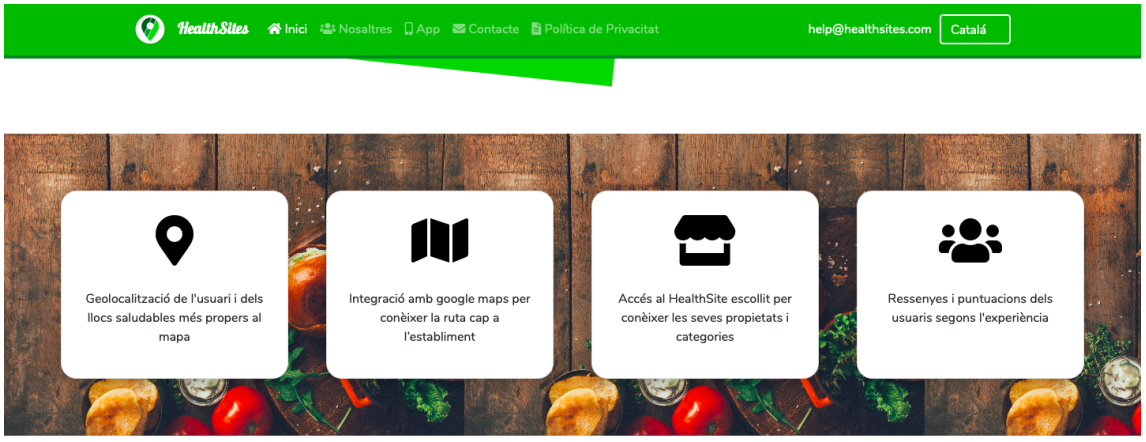


Figure 88: Inici part 2



Aprèn més

Qui som?

Coneix més sobre el nostre equip i com va començar aquest projecte

Figure 89: Inici part 3

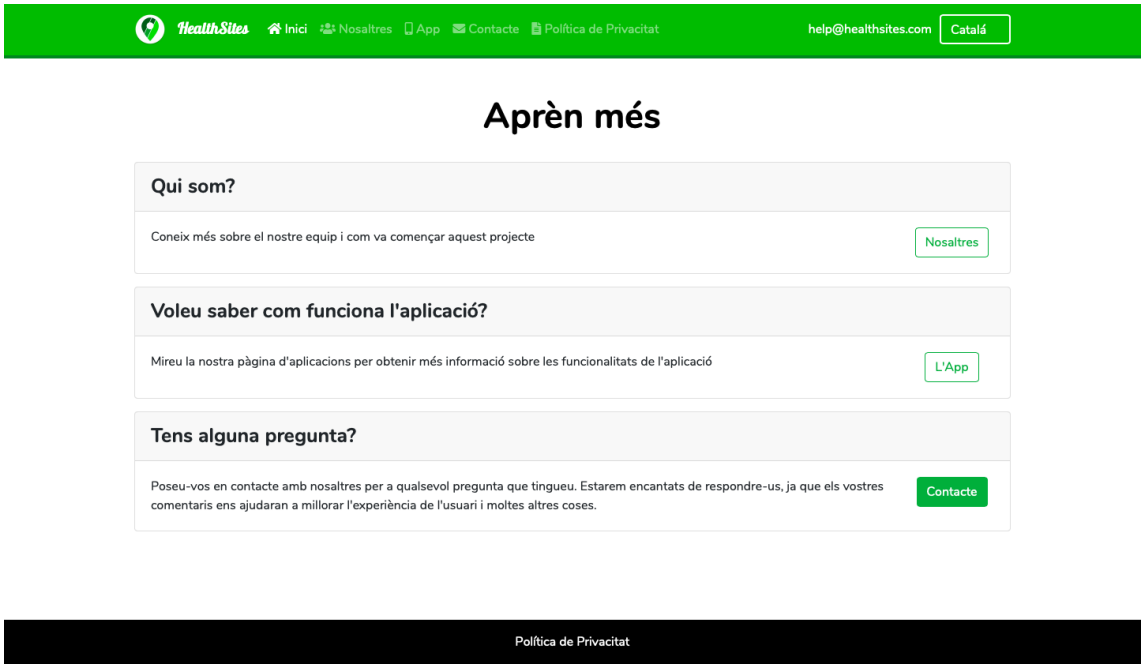


Figure 90: Inici part 4

Es va realitzar una pàgina de contacte per establir un altre canal a disposició dels usuaris i dels nouvinguts. D'aquesta manera poden contactar amb l'equip.

Pel que fa al seu desenvolupament intern, s'ha utilitzat un mòdul de "node.js" anomenat "nodemailer", el qual enllaça el formulari amb un servidor de correu mitjançant STMP (*Simple Transportation Management Protocol*).

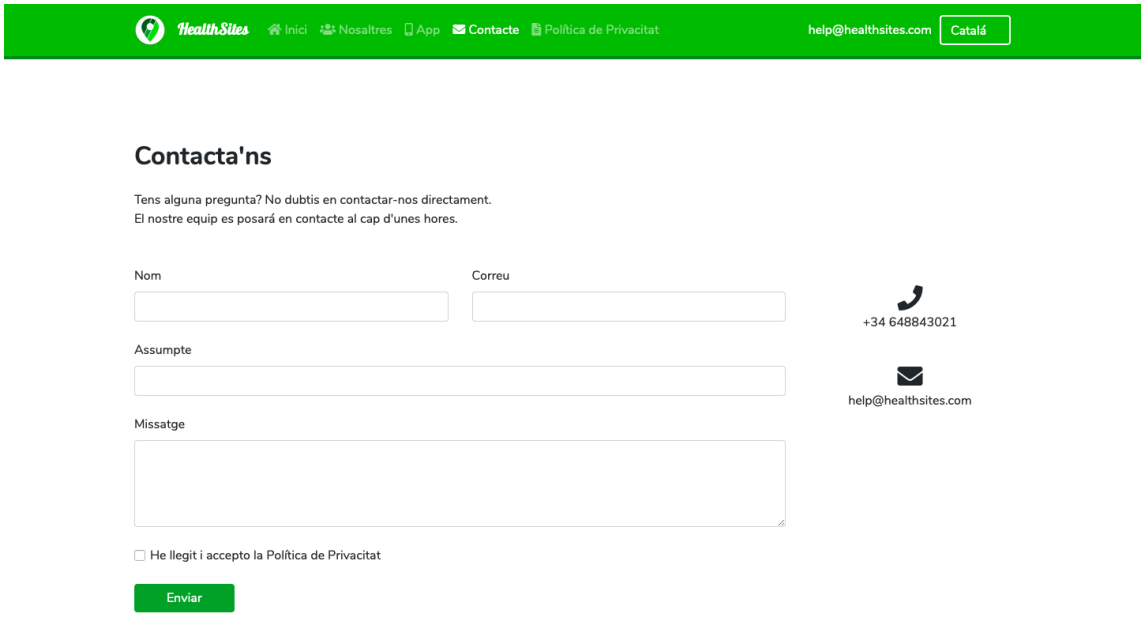


Figure 91: Pàgina de contacte de la web

Tot software requereix d'una política o termes d'ús. Per aquest motiu es va afegir, seguint diversos models web i en concordança amb la legislació vigent, una pàgina dedicada a la política de privacitat que la mateixa aplicació enllaça.

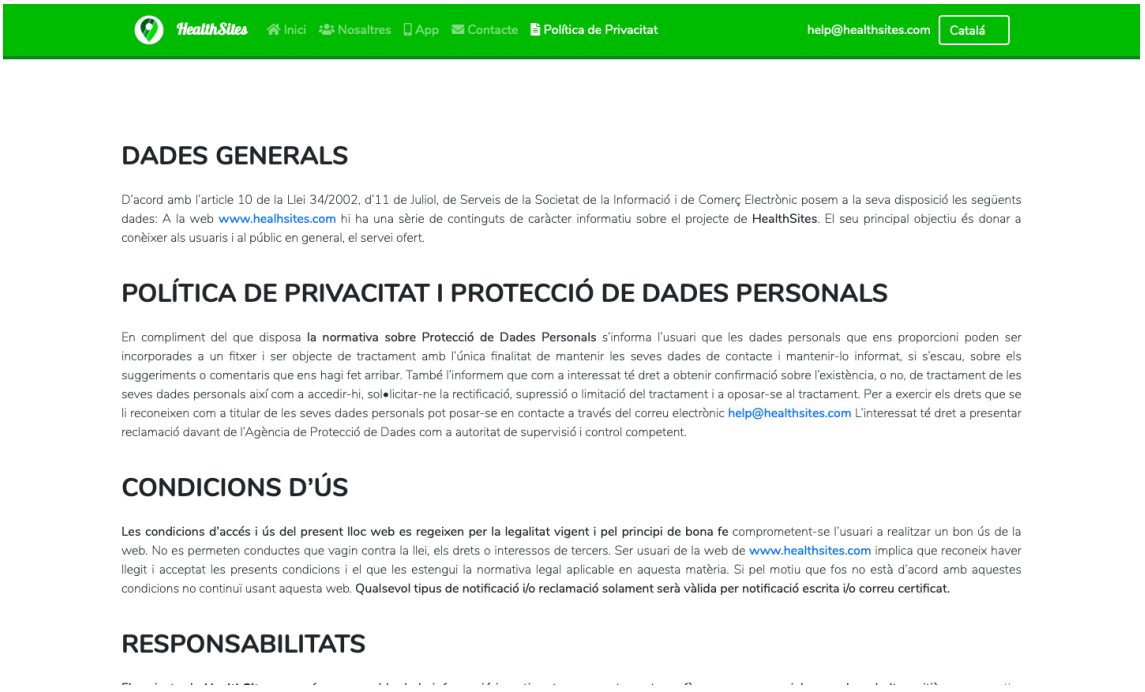


Figure 92: Pàgina de la política de privacitat de la web

9 Treball futur i Conclusions

9.1 Treball futur i possibles millores

Tot i que les bases ja estan plantades encara s'hi pot treballar moltíssim. Afegir noves funcionalitats, retocar pantalles específiques, etc. Justament continuar seguint la metodologia establerta des de l'inici.

Es podria millorar la interfície gràfica realitzant més proves amb usuaris ja que per les circumstàncies no es va poder. Seria ideal aconseguir un grup d'entre 15 a 20 persones per poder realitzar testos d'usabilitat i emprar tècniques com el *eye-tracking* molt útils per saber què s'aprecia més, menys o gens. Realment hi ha infinitat de proves per rebre *feedback* i aconseguir una interfície centrada en l'usuari.

Altrament, una de les parts més interessants es que es va plantejar crear una versió de l'aplicació per a les pròpies "HealthSites", en la qual els seus administradors puguin actualitzar les seves dades. I a més a més mantenir als clients actualitzats de la seva oferta, amb la capacitat d'afegir entrades, menús, imatges del local, etc. Tot mantenint la direcció d'interacció entre usuaris i "HealthSites".

I per la part de l'usuari normal tal com s'ha mencionat en seccions anteriors, aquest podria rebre notificacions "push" cada vegada que una "HealthSite", la qual segueix, actualitzi les seves dades o afegeixi una entrada nova, un menú, etc. Hi ha moltes possibilitats d'estendre la funcionalitat.

D'altra banda es podria crear una pantalla d'exploració, on es puguin veure les "HealthSites" properes. I així re-ubicar els filtres a la mateixa pantalla del mapa, on serien més accessibles; tot i que actualment i per la disposició del "bottom navigation" ja es va explicar la seva ubicació.

Es podria incorporar l'inici de sessió mitjançant *Google* i *Facebook*, per facilitar l'accés a l'aplicació i no haver d'obligar a l'usuari a utilitzar el seu correu electrònic.

També es podria desenvolupar la versió per a IOS i obrir-se al seu mercat. Es va plantejar l'opció de fer una conversió del .apk que seria una solució ràpida per a les primeres etapes del llançament de l'aplicació com a producte. Però realment si es vol evitar errors, s'ha de ser conscient que s'hauria de desenvolupar la versió respectiva.

Per acabar, es fa menció a certs aspectes molt interessants que podrien acabar de complementar la funcionalitat de l'aplicació.

9.1.0.1 Cloud Functions

Cloud Functions per *Firebase* és un *framework* "serverless" que permet executar de forma automàtica el codi de back-end en resposta a les sol·licituds HTTPS (*Hypertext Transfer Protocol Secure*).

Es desenvolupa codi *Javascript* o *TypeScript* i es guarda al núvol de *Google* per executar-se en un entorn ja administrat sense necessitat de controlar els servidors segons la demana per escalarlos o no. *Firebase* augmentarà els recursos de processament automàticament segons els patrons d'ús dels usuaris.

A més a més, per temes de seguretat, és un servei totalment aïllat de la part del client, en el nostre cas l'aplicació *Android*. Per tant és un servei fiable i segur ja que no podrem rebre atacs d'enginyeria inversa o modificacions al *front-end* gràcies a l'aïllament.

Les funcions escrites poden respondre a events generats per funcions de *Firebase* i *Google Cloud*; des d'activadors de *Firebase Authentication* fins a activadors de *Cloud Storage*. D'aquesta manera podem accedir als events i executar codi escalable com a resposta als mateixos.

Alguns casos pràctics són:

- Notifica als usuaris quan passa alguna cosa interessant
- Executa neteja i manteniment de la base de dades
- Executa tasques pesades al núvol enlloc de fer-ho a l'aplicació
- Realitza integracions amb APIs i Serveis externs

Es podria utilitzar per a infinitats de funcions, des d'enviar un email o notificació de benvinguda quan un usuari es registra, o enviar notificacions "push" segons certs events, entre molts altres.

9.1.0.2 Verificació de HealthSites

Una de les parts pendents del projecte és la verificació de les "HealthSites" com empreses reals. S'hauria de programar una tasca en el *back-end* que s'activés al moment d'afegir alguna "HealthSite".

En relació amb l'apartat de *Cloud Functions*, es podria usar aquest servei de *Firebase* per escriure la tasca de servidor adient segons l'event indicat, la qual hauria de verificar mitjançant un servei extern el CIF de l'empresa introduït.

Es va fer una recerca per tantejar una mica les possibilitats. No obstant es va trobar un *Web Service* de l'agència tributària espanyola per verificar CIFs d'empreses espanyoles, encara no s'està del tot segur si també seria vàlid per empreses d'altres països.

A part d'aquesta forma, també es va considerar realitzar una verificació del *Web Domain* de la "HealthSite". És a dir, el correu electrònic introduït com a contacte hauria de dependre del domini de la web. Aleshores s'enviaria un correu de verificació per assegurar l'autenticitat de les dades introduïdes.

9.2 Conclusions

Aquesta secció està dedicada a les conclusions finals i personals del projecte sobre l'aplicació "HealthSites", extretes al llarg del procés de desenvolupament del mateix.

S'han complert els objectius marcats inicialment. Principalment dissenyar i crear una aplicació *Android* totalment nova i actualitzada a partir d'una gran idea.

A nivell personal he profunditzat en el desenvolupament d'aplicacions per a dispositius *Android* i a gestionar completament un projecte de software d'aquesta magnitud de principi a fi. A causa de la situació viscuda aquest últim any hi ha hagut aspectes que no s'han pogut dur a terme del tot. Em refereixo a les validacions per part d'un grup d'usuaris o en realitzar més periòdicament reunions amb la dissenyadora on d'es d'aquí m'agradaria donar-li personalment les gràcies. Tot i que, també se'n pot treure coses positives perquè t'has de saber adaptar ja que de vegades les circumstàncies no seran les ideals

M'he endinsat en el món de *Kotlin*, que a nivell personal ha estat molt important i enriquidor, per tot el que comporta aprendre un nou llenguatge, la seva semàntica i possibilitats. Entendre com funciona i les avantatges respecte al desenvolupament tradicional, tals com reducció de línies de codi i pes de l'aplicació. També he adquirit molts coneixements sobre els nous serveis que ofereix la plataforma de *Firebase* i com utilitzar-los i sobretot integrar-los; i com analitzar i planificar un projecte en la seva totalitat.

Per acabar, també he aprés en organitzar i estructurar el codi, no fer-ho tot a la primera sinó pensar en com es pot reutilitzar aquest i millorar les relacions entre les diferents parts. I finalment, per tancar el projecte, aprendre com realitzar una pàgina web que recolzés en estil i continguts a l'aplicació.

Moltes Gràcies,

Albert Pérez Datsira